PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: WO 99/23585

(43) International Publication Date: 14 May 1999 (14.05.99)

(21) International Application Number: PCT/US98/23388

(22) International Filing Date: 3 November 1998 (03.11.98)

(30) Priority Data:
08/964,764 5 November 1997 (05.11.97) US

(71) Applicant: LEEP TECHNOLOGY, INC. [US/US]; Suite 101, 1210 Marina Village Parkway, Alameda, ca 94501 (US).

(72) Inventors: BAIR, John; 1931 Estabrook Way, Superior, CO 80027 (US). BENDER, Charles; 16 Amantes, Rancho Santa Margarita, CA 92688 (US).

(74) Agent: GOTLIEB, Charles, E.; Law Offices of Charles E. Gotlieb, Suite 300, 540 University Avenue, Palo Alto, CA 94301 (US).

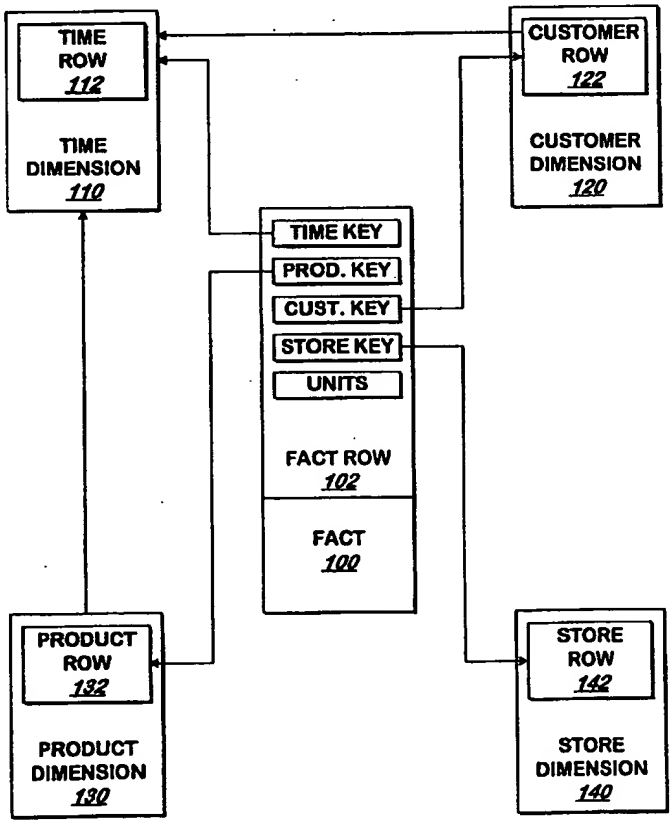
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR SELECTING ROWS FROM DIMENSIONAL DATABASES

(57) Abstract

A system and method selects rows from a fact table (100) in a dimensional database also containing a time dimension table (110) and other dimension tables (120, 130, 140). The other dimension tables each contain rows (122, 132, 142) containing a time invariant attribute to identify an item described by the row, an effective time attribute for the row, and other attributes. If an attribute for an item changes, a new row is added to the dimension table containing the time-invariant attribute for the item and current attributes for the item, without deleting or overwriting any existing rows for that item. One of the dimension tables (120, 130, 140) can be selected to create other tables using the time attributes existing in the dimension table. One of the resulting tables are then joined to a table in the dimensional database.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEM AND METHOD FOR SELECTING ROWS FROM DIMENSIONAL DATABASES

Express Mail Label Number

EI113835973US

Related Applications

The subject matter of this application is related to the subject matter of application serial number AA/AAA,AAA entitled, "Method and Apparatus for Producing Sequenced Queries" filed on July 25, 1997 by John Bair and Richard Snodgrass, having the same assignee as this application and is incorporated herein by reference in its entirety.

Field of the Invention

The present invention is related to database software and more specifically to database software for analyzing dimensional databases.

Background of the Invention

Databases are used to store information. The data in a database may be stored in several tables, with each table storing a subset of the information in the database. Each table in the database is divided into one or more rows, with each row containing related information and/or one or more pointers to information in one or more different tables. The pointers, as well as the information to which the pointer points, are known as "keys." The other information in a row is referred to as the "attributes" of the row. Attributes are sometimes referred to as "columns" of the table.

One example of a database table is a customer table containing information about the customers of a business. The customer table might contain one row for each customer of a business, with the attributes describing the name, address, birth date and marital status of each customer.

A conventional "dimensional database" is a database made of three or more tables arranged in a particular manner. A dimensional database contains a fact table and two or more dimension tables, with each row in the fact table pointing to one or more rows in one or more dimension tables. The fact table may contain the data that is expected to change most rapidly, with the dimension tables storing data that changes less rapidly. For example, in a database that records orders for products, an order described in each row of the fact table might contain a pointer to the customer row in the customer table corresponding to the customer who placed the order. Each row in the fact table may also contain a key to a row in a product dimension table corresponding to the product in the order. Each row in the fact table may also contain a key to a row in a store dimension table corresponding to the store at which the order was placed. Additionally, each row in the fact table may contain a key to a row in a time dimension table corresponding to the date and time the order was placed. The "fact" recorded by the fact table is that an order was placed for a specific customer containing a specified product at a particular store at a stated time.

Referring now to Figure **1A**, such a dimensional database is illustrated. Fact table **100** stores fact records **102**, each containing a time key, a product key, a customer key, and a store key described below. In addition, attributes containing the number of units of the product ordered are stored in the fact row **102**.

A customer dimension table **120** stores one or more customer rows **122**, containing information about each of the customers that may place orders for products using the fact table **100**. Store dimension table **140** holds one or more store rows **142**, containing information about the stores that may take orders for products. Product dimension table **130** contains one or more product rows **132**, which hold information about each of the products that may be ordered. Each of the rows of the tables **120, 130, 140** contains a computer-generated integer key, or other form of key, unique within the table **120, 130, 140**. To point to a row of one of these tables **120, 130, 140**, the corresponding key in the fact table is set to the value of the key of the row.

Time dimension table **110** stores one or more time rows **112**, each of which contains a key and an identifier of the date and time to which the index corresponds. For example, each time row **112** may contain a unique, computer-generated integer key or other form of key, and a date and time to a precision of one minute. The time dimension table **110** may be set up in advance to contain rows **112** corresponding to all possible values of time past, present, and future for a limited amount of time, such as a 10 year period.

The time key, product key, customer key, and store key in each row **102** of the fact table **100** are pointers to a row **112, 122, 132, 142** in the time dimension table **110**, customer dimension table **120**, product dimension table **130** or store dimension table **140**, respectively.

A dimensional database may be implemented using a conventional relational database program such as the Oracle7.3 product commercially available from Oracle Corporation of Redwood Shores, California or the Microsoft Access product commercially available from Microsoft Corporation of Redmond, Washington. View or Virtual databases may also be used, treating several databases as if

they were a single database. Conventional relational databases with specialized tools for On-Line Analytical Processing, or OLAP-optimized databases may also be used. Such databases are sometimes referred to as MOLAP, ROLAP or DOLAP databases and are described at http://www.sentrytech.com/dw05dem.htm. Non-database implementations such as those storing data using objects, records, arrays or flat files may be used to implement dimensional databases. Keys may be implemented using conventional pointers or look-up table approaches.

A database program may generate some of the keys in fact row **102** automatically, with other keys selected by the person or program operating the database program. For example, when an order is placed, a new fact row **102** is generated in the fact table **100** by a person using, or a computer program interacting with, a database program. At this time, the time key may be selected by locating a time row in time dimension table **110** that most closely matches the system clock of the computer system on which the database program is running. The database program will write the index of this time row **112** to the time key in the fact row **102**. The store key may be pre-filled in to correspond to the store row **142** in the store dimension table **140** corresponding to the store at which the order is being placed. The person or program interacting with the database program adds the customer key to the fact row by selecting a row in the customer dimension table **120** corresponding to a name or customer number provided by the customer.

The person or program then selects a product from the product dimension table **130** and signals the database program to enter the corresponding product key of the corresponding product row **132**. The person or program then selects the number of units for the product designated by a product key which is stored in the fact row, thereby completing the order

in the fact table **100**.

Conventional dimension tables **110**, **120**, **130**, **140** may change quite slowly in relation to the fact table **100**. For example, a customer row **122** may contain fields for marital status and address. The customer table **120** may only change when a customer is added, or an existing customer changes his or her name, marital status or address, while many orders are expected each hour in the fact table **100**.

Conventional dimensional databases may be implemented with a temporal fact table **100**. A "temporal table" is a table that maintains historical information in addition to current information.

Unlike the fact table **100**, many conventional dimensional database dimension tables **120**, **130**, **140** are not temporal tables. If a customer changes his or her address or gets married or divorced, the old information in the customer dimension table is replaced with current information. Because the dimension tables **120**, **130**, **140** are not temporal, changes to the data in these tables **120**, **130**, **140** can produce inaccurate results when the database is later queried. For example, if a customer changes his or her marital status, the database program may simply update the customer row **122** for that customer in the customer dimension table **120**, deleting the old value of the marital status attribute and replacing it with the new value. Because the customer's marital status is not preserved over time, queries to the dimensional database **100** that depend on time of a change or an effective date of a customer's marital status will be inaccurate. For example, if a customer places an order while the customer is single, and then changes his or her marital status to "married", querying the fact table **100** for orders placed by single customers will omit the fact row **102** corresponding to the order placed when the customer was single.

Several solutions have been attempted to record changes

in data related to the dimension tables **120**, **130**, **140** without causing those tables to become excessively large in size. For example, another row in the customer dimension table **120** may be added each time a change is made to the information for customer. Subsequent orders made by the customer point to the new customer row **122**. However, such solutions have not recorded the time of the change. Because only the rows of the fact table **100** record any indication of when a change is made, when a query is performed on the database, the first order to be placed by that customer after the change was made may erroneously be identified as the best indication of the time of that change. Such inaccuracy prevents accurate queries from being made to the database, such as a list of all orders placed within one month of a change of a marital status.

Another attempted solution is to add storage for an old value of an attribute for some or all of the attributes stored by each row **122**, **132**, **142** in the dimension tables **120**, **130**, **140**. Such an arrangement allows an attribute to be updated, while preserving the old value of the attribute. However, this attempted solution does not store more than one older value of each attribute, does not store the time of the change and may require re-programming of the applications accessing the data in the database to implement. Such an attempted solution will not allow queries to be made to the database that depend on the time of the change or effective date of a value of an attribute.

A system and method are therefore necessary that can accurately select records from a dimensional database that depend on a time of a change or effective date of a value of an attribute in a table of a dimensional database.

Summary of Invention

A system and method selects records from a fact table of a dimensional database. The dimensional database contains the fact table, a time dimension table and one or more other dimension tables. Each of the other dimension tables contain rows, with each row containing a time invariant attribute that relates the row to the item the row describes, one or more effective time attributes that describe the effective time of the data in the row, and other attributes. To change an attribute in one of these other dimension tables, a row is added to the table with the time invariant attribute of the item being described, and the other attributes are updated as necessary. The effective time attribute or attributes of the new row, and optionally an ending effective time attribute of the most recent historical row, are updated so that the table reflects the effective time of the new and historical rows.

Temporal query primitive functions may then be applied to the dimension tables. Temporal query primitive functions select rows from dimension tables, or create new tables, based upon one or more of the effective time attributes, other attributes of the dimension tables or both. The resulting tables may then be joined to one or more of the dimensional database tables to select rows from one or more of the dimensional database tables.

Brief Description of the Drawings

Figure **1A** is a block diagram illustrating a conventional dimensional database.

Figure **1B** is a block schematic diagram of a conventional computer system.

Figure **1C** is a block diagram illustrating a dimensional database according to one embodiment of the present invention.

Figure **2** is a block diagram illustrating a dimension record according to one embodiment of the present invention.

Figure **3A** is a flowchart illustrating a method of performing one or more temporal query primitive functions and one or more joins on the tables of a dimensional database according to one embodiment of the present invention.

Figure **3B** is a flowchart illustrating a method of performing a join on the tables of a dimensional database according to one embodiment of the present invention.

Figure **4A** is a flowchart illustrating a method of performing a temporal projection on a table of a dimensional database that contains a beginning effective time attribute and an ending effective time attribute according to one embodiment of the present invention.

Figure **4B1** is a flowchart illustrating a method of performing a temporal projection on a table of a dimensional database that contains a beginning effective time attribute and a status attribute according to one embodiment of the present invention.

Figure **4B2** is a flowchart illustrating a method of performing a temporal projection on a table of a dimensional database that contains a beginning effective time attribute and a status attribute according to an alternate embodiment of the present invention.

Figure **4B3** is a flowchart illustrating a method of performing a temporal projection on a table of a dimensional database that contains a beginning effective time attribute and a status attribute according to an alternate embodiment of the present invention.

Figure **4C** is a flowchart illustrating a method of performing a temporal projection on a table of a dimensional

database that contains a beginning effective time attribute according to one embodiment of the present invention.

Figure **4D** is a flowchart illustrating a method of selecting a temporal projection to perform according to one embodiment of the present invention.

Figure **5** is a flowchart illustrating a method of performing a temporal selection on a table in a dimensional database according to one embodiment of the present invention.

Figure **6A** is a flowchart illustrating a method of performing a temporal normalization on a table in a dimensional database containing a beginning effective time attribute, an ending effective time attribute according to one embodiment of the present invention.

Figure **6B** is a flowchart illustrating a method of performing a temporal normalization on a table in a dimensional database containing a beginning effective time attribute according to one embodiment of the present invention.

Figure **6C** is a flowchart illustrating a method of performing a temporal normalization on a table in a dimensional database containing a beginning effective time attribute and a status attribute according to one embodiment of the present invention.

Figure **6D** is a flowchart illustrating a method of selecting a temporal normalization to perform according to one embodiment of the present invention.

Figure **7A** is a flowchart illustrating a method of performing a state duration identification on a table of a dimensional database according to one embodiment of the present invention.

Figure **7B** is a flowchart illustrating a method of performing a temporal normalization and a state duration identification on a table of a dimensional database according to one embodiment of the present invention.

Figure **8A** is a flowchart illustrating a method of performing a transition detection on a table in a dimensional database according to one embodiment of the present invention.

Figure **8B** is a flowchart illustrating a method of performing a transition detection on a table in a dimensional database according to an alternate embodiment of the present invention.

Figure **9** is a block schematic diagram of an apparatus for performing one or more temporal query primitive functions on one or more tables in a dimensional database and joining the result of temporal query primitive functions or results derived therefrom with other tables of the dimensional database according to one embodiment of the present invention.

Detailed Description of a Preferred Embodiment

The present invention may be implemented as computer software on a conventional computer system. Referring now to Figure **1B**, a conventional computer system **150** for practicing the present invention is shown. Processor **160** retrieves and executes software instructions stored in storage **162** such as memory, which may be Random Access Memory (RAM) and may control other components to perform the present invention. Storage **162** may be used to store program instructions or data or both. Storage **164**, such as a computer disk drive or other nonvolatile storage, may provide storage of data or program instructions. In one embodiment, storage **164** provides longer term storage of instructions and data, with storage **162**

providing storage for data or instructions that may only be required for a shorter time than that of storage **164**. Input device **166** such as a computer keyboard or mouse or both allows user input to the system **150**. Output **168**, such as a display or printer, allows the system to provide information such as instructions, data or other information to the user of the system **150**. Storage input device **170** such as a conventional floppy disk drive or CD-ROM drive accepts via input **172** computer program products **174** such as a conventional floppy disk or CD-ROM or other nonvolatile storage media that may be used to transport computer instructions or data to the system **150**. Computer program product **174** has encoded thereon computer readable program code devices **176**, such as magnetic charges in the case of a floppy disk or optical encodings in the case of a CD-ROM which are encoded as program instructions, data or both to configure the computer system **150** to operate as described below.

In one embodiment, each computer system **150** is a conventional Compaq Deskpro 6000 computer running the Windows NT operating system commercially available from Microsoft Corporation of Redmond, Washington, although other systems may be used.

Referring now to Figure **1C**, a dimensional database arranged according to one embodiment of the present invention is shown. Any of the implementations of a dimensional database described above may be used to implement the present invention. The dimensional database is arranged similar to that shown in Figure **1A**. However some of the dimension tables **120, 130** are joined to the time dimension table **110**. The time dimension table **110** has a resolution equal to the smallest resolution required by the tables **100, 120, 130** joined to it. In one embodiment of the present invention, all

dimension tables **120, 130, 140** are joined to the time dimension table **110** except the time dimension table **110** itself.

Referring now to Figure **2**, a record for use as a row in a dimension table or fact table is shown according to one embodiment of the present invention. The record shown in Figure **2** would ordinarily not be used as a row in a time dimension table, although the present invention will operate with such rows in the time dimension table. As used herein, the term "record" denotes the collection of data that when inserted into a table, will become a row of the table.

A time invariant attribute **210** is assigned to the item described by the record **230**. The time invariant attribute provides a reference to a single item. For example, if the dimension record **230** is a customer record, the time invariant attribute may be customer number assigned by the database program. The time invariant attribute **210** is unique to the item described by the record **230**, however, as described below the time invariant attribute **210** may be duplicated in other records which describe the same item.

A beginning effective time attribute **212** is also a part of the dimension record **230**. In one embodiment, beginning effective time attribute **212** holds a key to a time row in the time dimension table. Beginning effective time attribute **212** describes the time at which the record **230** begins its effectiveness. For example, beginning effective time attribute might describe the day on which a customer changed his or her marital status. An ending effective time attribute (not shown) may also be a part of the record as described below, or it can be inferred using the beginning effective time attribute **212** of the row having the next highest beginning effective time attribute and the same time invariant attribute.

In an alternative embodiment of the present invention,

attribute **212** is an ending effective time attribute. In such embodiment, the beginning effective time may be inferred using the ending effective time of the record having an ending effective time attribute **212** that precedes most closely the ending effective time attribute **212** of the row **230** and contains the same time invariant attribute **210**.

In one embodiment, if an ending effective time attribute is used in a dimension record, the ending effective time attribute references the first time row in the time dimension table on or after the actual ending effective time of the record. In another embodiment, the ending effective time attribute references the last time row in the time dimension table on or before the actual ending time of the record. As long as any applications which use the information in the record correctly handle the ending effective time, either approach may be used, although it can simplify programming of such applications if the approach is consistently used across all tables of the database.

An optional surrogate attribute **214** may also be used. The surrogate attribute **214** may be used as a key to the table in which the record is inserted. As such, each record contains a value different from the surrogate attribute of the rows of the table into which it will be inserted. In one embodiment, the surrogate attribute is assigned a value that is a function of the time invariant attribute **210** and the beginning effective time attribute **212**. The function may be a concatenation of the two attributes in one embodiment. If no surrogate attribute is used, the time invariant attribute and the beginning effective time attribute may be used as a key to the table into which the record will be inserted. However, some databases do not perform joins as efficiently using two attributes as a key, and therefore, the use of a surrogate attribute can be more efficient.

One or more time variant attributes **216** may be stored as

part of the dimension record **230**. Time variant attributes are any attributes for an item that will change over time, for example a customer's name, address and marital status.

An optional status attribute **218** is also stored as a part of the dimension row **230**. The status attribute **218** has a value of null if the item described by the dimension record **230** is still considered to be part of the database. The status attribute **218** has a non-null value indicating 'terminated' if the item described by the dimension record **230** is not considered to be active in the database. For example, if a person ceases being a customer, the status attribute is set to 'terminated' in a dimension record to be inserted as a row in the customer table with the beginning effective time attribute **212** describing the date person ceased to be a customer and the time invariant attribute equal to the time invariant attribute used in the other rows describing that customer.

If an attribute for an item changes, the current row for the item remains in the database table unchanged. In addition, a new record is created and added as a new row to the table. The new row allows all the old values of the unchanged attributes to be maintained, and the beginning effective time attribute **212** of the new row or the ending effective time of the most recent row preceding the new row maintains the effective time of change for use as described below.

Referring now to Figure **3A**, a method of analyzing a dimensional database is shown according to one embodiment of the present invention. One or more temporal queries are received **308**. A "temporal query" according to the present invention contains at least one statement to perform one or more temporal query primitive functions described in more detail below and at least one statement to perform one or more joins described in more detail below. The temporal

query may optionally contain statements to perform conventional selections such as SQL selections, and may contain one or more statements to build one or more intermediate tables described in more detail below. If any non-temporal selections or projections are received, they are performed **310**. Non temporal selections or projections are conventional selections or projections performed on one or more tables. Step **310** may be performed to reduce the number of rows in the tables in the dimensional database. Step **310** may be performed either before (as shown in the figure) or after step **318** or some selections may be performed before step **318** and some performed after step **318**. In one embodiment, the choice of performing any non temporal selections before step **318** described below is made to cause the most restrictive operation to be performed first, thereby allowing the other operation to be performed more efficiently.

As described in more detail below, a temporal query primitive function uses a time attribute of the rows in a table to either select rows from an existing table, thereby reducing the number of rows that are selected in a table, or create a new table from an existing table, in such a manner that the new table has fewer rows than the existing table. The name of the existing table on which to perform the temporal query is received as a part of step **308**. One of the query primitive identifiers received in step **310** is chosen **312**. In one embodiment, step **312** is performed using conventional query parsing techniques, with operations performed in the order of the statement containing them, with innermost operations selected first.

The temporal query primitive functions described below may be performed on a table sorted as described herein. If the table corresponding to the query primitive identifier chosen in step **312** is not sorted **314**, the existing table

having the name specified in the query primitive chosen at step **312** is sorted **316**. In one embodiment, the sort is specified as part of the temporal query received in step **308**. The sort is performed in such a manner that all of the rows in the existing table having the same time invariant attribute are adjacent to one another, and that rows of the existing table with the same time invariant attribute are ordered in ascending order of the beginning effective time attribute or ending effective time attribute. For example, the existing table can be sorted in ascending order of time invariant attribute, with ties broken in order of ascending beginning effective time attribute. In one embodiment, the effective time attributes are integer indexes to the effective time, and are assumed to have the same relative order as the values of the times or dates corresponding to the keys. It is also possible to sort rows with the same time invariant attributes in descending order, and adjust the details of the method of the present invention accordingly.

In one embodiment, the table named in the temporal query primitive selected in step **312** is not physically sorted. Instead, an index is maintained which logically orders each row of the table as described above. The temporal query primitive functions described below operate on the rows of the table in the order of the index.

As described in more detail below, a temporal query primitive function corresponding to one of the statements received as a part of the query received in step **310** and chosen in step **312** is performed **318** on the fact table or one or more dimension tables of a dimensional database, either as originally provided or as selected in step **308**. If additional temporal query primitive functions were received **320** in step **312**, the method continues at step **312** by selecting a different temporal query primitive identifier until all of the temporal query primitives received in step

310 are performed.

If the temporal query received in step 308 contain statements that cause intermediate tables to be built, described below, the intermediate tables are built 321. As described in more detail below, intermediate tables are tables built using the result of a temporal query primitive performed in step 318, either by conventional selection, projection or join techniques.

One or more tables are then joined 322 to other tables as described in more detail below. The tables joined in step 322 can be any one or more tables from the dimensional database, as it existed originally, or as selected in step 310, a result of a query primitive performed in step 318, an intermediate table built in step 321 or the result of a join performed in step 322.

Referring now to Figure 3B, a method of joining a first table with a second table is shown according to one embodiment of the present invention. Figure 3B illustrates step 322 of Figure 3A. It can be helpful to understand the structure of a database table to understand how the method of Figure 3B operates.

A database table may have certain rows that are "selected" and other rows that are not selected. The selected rows may be distinguished from the unselected rows in a variety of ways. In one embodiment, each row in the table is pointed to using arrays or dynamic arrays, with one array for each index of the table. Two ways of selecting rows in a table are possible. The first way creates a new table name. If rows of the table are selected, new arrays in the order of the old arrays but pointing only to the selected rows of the table may be generated and references to the new table name will use the new arrays. Another way of selecting rows in a table is to maintain a separate "unselected" array containing pointers to the rows in the table not

corresponding to the selection criteria. Rows of a table not selected are removed from the array corresponding to the table name and placed into the unselected array. References to the table name do not include the rows on the unselected array.

To join a first table with one or more second tables, a row counter is initialized to a value of '1' to designate a "current" row in the database **350**. If the first table contains a reference to a row in a second table that is not selected **352**, the row in the first table is unselected **356** as described above, and otherwise the row is selected **354**. If there are more rows in the table **358**, the row counter is incremented **360** and the method continues at step **352**. Otherwise, the method terminates **362**.

In addition to the join described above with reference to Figure **3B**, the present invention can utilize joins implemented using other methods such as Cartesian product joins or sort joins. These joins are described in Mannila & Raiha, "The Design of Relational Databases," (Addison Wesley, 1992, ISBN 0-201-56523-4).

Temporal Query Primitive Functions.

In one embodiment, temporal query primitive functions performed in step **318** of Figure **3A** include temporal projection, temporal selection, temporal normalization, state duration, transition detection and any other conventional temporal query functions. Temporal projection, temporal selection, temporal normalization, state duration and transition detection are described below.

Temporal Projection.

Temporal projection creates from an input table, an output table describing the effective start and stop times of the rows in the input table. The input table may be a fact table, dimension table, a result of a different temporal query primitive, the result of a join, any tables selected

from such tables or any other table in which each row has some or all of the attributes in the record described above with reference to Figure 2. The output table created using temporal projection has the same number of rows as the input table. The output table contains two attributes, also referred to as "columns", a beginning effective time, and an ending effective time. Temporal projection may be performed according to the structure of the rows in the input table.

Referring now to Figure 4A, a method of implementing temporal projection is shown according to one embodiment of the present invention. The method shown in Figure 4A is used in one embodiment if the rows in the input table contain both beginning and ending effective time attributes. The table names to be used as the input and output tables are received 408. A row counter, used to designate a row in the input and output tables as the "current row" is initialized to a value of '1' 410. A record to be inserted as a row in the output table is created 412. The beginning effective time and ending effective time attributes from the row in the input table corresponding to the row counter are copied 414, 416 into the beginning effective time and ending effective time columns of the record created in step 412. The record created in step 412 and modified as described in Figure 4A is then appended 418 as a row at the end of the output table. As used herein, "append" can include creating a table if necessary, and inserting the record into the end or elsewhere in the table. If at least one additional row exists in the input table past the row corresponding to the row counter 420, the row counter is incremented 422 and the method continues at step 412. Otherwise, the method terminates 424.

Referring now to Figure 4B1, a method of implementing temporal projection is shown according to another embodiment of the present invention. The method shown in Figure 4B1 is used in one embodiment if the rows in the input table contain

both beginning effective time and status attributes. The names of the input and output tables are received **428**. A row counter is initialized to a value of '1' **430**. A record is created for use in the output table **432** having the same structure as the row created in step **412** of Figure **4A**. If the row in the input table corresponding to the row counter has a status attribute indicating the row has been terminated, both attributes in the record created in step **432** are set to null **434**, **436**, and the method continues at step **446**. Otherwise, the beginning effective time from the row in the input table corresponding to the row counter is copied **438** into the beginning effective time column of the record created in step **432**.

If a row exists that follows the row in the input table corresponding to the row counter and contains the same time invariant attribute as the row in the input table corresponding to the row counter **440**, the beginning effective time attribute from the row following the row in the input table corresponding to the row counter is copied **442** and inserted as the ending effective time attribute of the record created in step **432**. If such a row does not exist, an indicator corresponding to value of "now" is inserted **444** into the ending effective time attribute of the record for the output table created in step **432**.

The record created in step **432** and modified as described in Figure **4B1** is appended **446** to the end of the output table. If more rows exist in the input table following the row corresponding to the row counter **448**, the row counter is incremented **450** and the method continues at step **432**. Otherwise, the method terminates **452**.

In another embodiment of the present invention shown in Figure **4B2**, the record created in step **432** is discarded **436** if the row in the input table corresponding to the row counter has a status attribute indicating the row has been

terminated **434**, and the method continues at step **448**. In such embodiment, the number of rows in the output table will not equal the number of rows in the input table, making correlation between the rows of the input table and the rows of the output table more difficult. If desired, each of the records created in step **430** may be created with an extra attribute storing the row counter, or a key to the corresponding row of the input table, such as the surrogate key, may be copied into the record created in step **432** at step **438**. The other steps shown in Figure **4B2** operate as described above with respect to Figure **4B1**.

In still another embodiment of the present invention shown in Figure **4B3**, if the row in the input table corresponding to the row counter has a status attribute indicating the row has been terminated **434**, if another row exists in the input table **436**, the row counter is incremented **437** and the method continues at step **434**, and if another row does not exist in the input table, the method terminates **452**. The other steps shown in Figure **4B3** operate as described above with respect to Figure **4B1**. The alternative embodiments of the method shown in Figures **4B2** or **4B3** are used in one embodiment if the rows in the input table contain both beginning effective time and status attributes.

Referring now to Figure **4C**, a method of implementing temporal projection is shown according to another embodiment of the present invention. The method shown in Figure **4C** is used in one embodiment if the rows in the input table contain a beginning effective time attribute, but no status or ending effective time attribute.

The input and output table names are received **468**. A row counter is initialized to a value of '1' **470**. A record is created for use in the output table **472** having the same structure as the row created in step **412** of Figure **4A**. The beginning effective time of the row in the input table

specified in step **468** corresponding to the row counter is copied **474** into the row created in step **472**. If a row exists in the input table following the row corresponding to the row counter and that row has a time invariant attribute equal to the time invariant attribute of the row in the input table corresponding to the row counter **476**, the beginning effective time from the row following the row in the input table corresponding to the row counter is copied **478** into the ending effective time attribute of the record created in step **472**. Otherwise, a value denoting "now" is inserted **480** into the ending effective time attribute of the record created in step **472**. The record created in step **472** and modified as described above is appended **482** as a row to the end of the output table specified in step **468**. If there are rows in the input table following the row corresponding to the row counter **484**, the row counter is incremented **486** and the method continues at step **472**. Otherwise, the method terminates **488**.

In one embodiment, the names or other identifiers of the beginning effective time attribute, ending effective time attribute or status attribute are received in addition to the names of the input and output tables. Selection from among the methods of performing temporal projection depends on which names are received. Referring now to Figure **4D**, a method of selecting a method of performing a temporal projection is shown according to one embodiment of the present invention. The names of the input and output tables are received **489**. The names of the beginning effective time attribute, ending effective time attribute or status attribute are also received **490**. In one embodiment, the names of any of these attributes that exist in the table are received as part of step **490** and the other names are omitted. If, in step **490**, the names of both beginning and ending effective time attributes are received **491**, the method

described above with reference to Figure **4A** is used to perform the temporal projection **492**. If, in step **490**, the names of the beginning effective time attribute and status attribute are received **493**, one of the methods described above with reference to Figures **4B1**, **4B2** or **4B3** is used to perform the temporal projection **494**. If, in step **490**, the name of the beginning effective time attribute is received **495**, the method described above with reference to Figure **4C** is used to perform the temporal projection **496**. Otherwise, the method terminates **497**.

Temporal Selection.

Temporal selection may be performed on an input table that is either a dimension table, a fact table or any other table with each row having some or all of the attributes of the record described above with reference to Figure **2**. A method of temporal selection receives the name or other identifier of an input table, the name or other identifier of a temporal projection table corresponding to the input table and a Boolean function. A "temporal projection table" is an output table created by a temporal projection as described above. The Boolean function is applied to each row of the temporal projection table. If the function as applied is true, the corresponding row in the input table is selected. Corresponding rows in the input table for which the function is not true in the temporal projection table are not selected.

Referring now to Figure **5**, a method of performing temporal selection is shown according to one embodiment of the present invention. The function to be used for the temporal selection, the name or other identifier of the input table and the name or other identifier of the temporal projection table corresponding to the input table are received **508**. A row counter is initialized to a value of '1' **510**.

The function is applied to the row of the temporal projection table corresponding to the row counter and if the result of the function is true **512**, the row in the input table corresponding to the row counter is selected **514** using any selection method described above. Otherwise, the row in the input table corresponding to the row counter is not selected **516** using any of the selection methods described above. If there are more rows in the temporal projection table beyond the row corresponding to the row counter **518**, the row counter is incremented **520** and the method continues at step **512**. Otherwise, the method terminates **522**.

Temporal Normalization.

Temporal normalization creates an output table that is a consolidated version of an input table. The input table can be a fact table, a dimension table, or any other table in which each row has some or all of the attributes of the record described above with reference to Figure **2**. Two or more rows of the input table having the same value of selected attributes and the same time invariant attribute are consolidated to a single row of the output table, provided the effective time attributes of such rows of the input table are contiguous in time. Rows are contiguous in time if no gaps exist between the ending effective time of one row and the beginning effective time of the row that is next in time. Temporal normalization may be performed differently depending on whether a status attribute exists in the input table, and whether an ending effective time attribute exists in the input table.

Referring now to Figure **6A**, a method of temporal normalization is shown according to one embodiment of the present invention. The method of Figure **6A** is used if the input table does not contain a status attribute and does contain beginning and ending effective time attributes. The input and output table names, and identifiers of the

attributes on which to normalize the input table are received **608**. A row counter is initialized to a value of '1' **610**. A record to be used as a row for the output table is created **612**. In one embodiment, the record created contains space for a time invariant attribute, beginning and ending effective time attributes, and the selected attributes identified in step **608**. Other attributes from the input table may also be a part of the record created in step **612**.

The beginning and ending effective time attributes are copied **614** from the row in the input table corresponding to the row counter to the record created for the output table in step **612**. In addition, the time invariant attribute and other attributes such as the selected attributes identified in step **608** are also copied from the same record of the input table to the record created in step **612**.

The row counter is incremented **616**. The row in the input table corresponding to the row counter is compared to the record created in step **612** as modified in step **614**. If such a row exists **618**, and the invariant attribute of the row in the input table corresponding to the row counter matches **620** the invariant attribute copied in step **614**, and the beginning effective time attribute of the row in the input table corresponding to the row counter identifies the row corresponding to the row counter as contiguous in time to the ending effective time attribute copied in step **614**, for example by matching, and the attributes corresponding to those identified in step **608** from the row in the input table corresponding to the row counter match **624** those copied in step **614**, the ending effective time attribute of the row in the input table corresponding to the row counter is copied **626** into the ending effective time attribute of the row created for the output table in step **612**, and the method continues at step **616**. Otherwise, the record created in step **612** and modified as described in the Figure **6A** is appended as

a row at the end of the output table **628**. If the input table contains a row corresponding to the value of the row counter **630**, the method continues at step **612**. Otherwise, the method terminates **632**.

If the input table does not contain an ending effective time attribute, the method described above with reference to Figure **6A** is used, however step **622** is omitted, and steps **614** and **626** do not copy the ending effective time attribute to the row created in step **612**. This embodiment is illustrated in Figure **6B** with the remaining steps operating as described with reference to Figure **6A**.

If the input table does not contain an ending effective time attribute and instead contains a status attribute, the method may be altered to accommodate this change in structure. Referring now to Figure **6C**, a method of performing temporal normalization is shown according to an alternate embodiment of the present invention. Steps **608**, **610**, **612**, **614**, **616**, **618**, **620**, **624**, **628**, **630** and **632** operate as described above with reference to Figure **6B**. Step **622A** verifies that the status of the row in the input table corresponding to the row counter is not terminated, and if the status is terminated, the method continues at step **628**. In addition, steps **634** and **636** are added between steps **628** and **630**. Step **634** identifies whether the status of the row in the input table corresponding to the row counter is terminated. If so, the row counter is incremented **636**. Otherwise, the method continues at step **630**.

In one embodiment, the names of the beginning effective time attribute, ending effective time attribute or status attribute, if such attributes exist, are received in addition to the names of the input and output tables and the selected attribute identifiers. Selection from among the methods of performing temporal normalization described above depends on which of the beginning effective time, ending effective time

or status attribute names are received. Referring now to Figure 6D, a method of selecting a method of performing a temporal projection is shown according to one embodiment of the present invention. The names of the input and output tables are received 680. The names of the selected attributes and the beginning effective time attribute, ending effective time attribute or status attribute are also received 682. In one embodiment, the names of any of these attributes that exist in the input table are received with the other attribute names omitted. If, in step 682, the names of both beginning and ending effective time attributes are received without the name of a status attribute 684, the method described above with reference to Figure 6A is used to perform the temporal normalization 686. If, in step 682, the name of the beginning effective time attribute is received without the names of the ending effective time or status attribute 688, the method described above with reference to Figure 6B is used to perform the temporal normalization 690. If, in step 682, the name of the beginning effective time attribute is received with the name of the status attribute 692, the method described above with reference to Figure 6C is used to perform the temporal projection 694. Otherwise, the method terminates 696.

State Duration.

The temporal primitive function of state duration creates in an output table one row for each row in an input table. The input table may be a fact table, a dimension table or any other table in which each row has some or all of the attributes in the record described above with reference to Figure 2. Each row in the output table contains a single attribute, containing the duration of the corresponding row of the input table. The input table may be any table that contains a beginning and ending effective time attribute, such as a temporal projection table. Thus, state duration

may be applied to a dimension table by first performing temporal projection on the dimension table, and then performing state duration on the output of the temporal projection.

Referring now to Figure **7A**, a method of performing state duration is shown according to one embodiment of the present invention. The name or other identifier of the input and output tables are received **708**. A row counter is initialized to a value of '1' **710**. A record is created **710** for the output table containing a duration attribute. The beginning and ending effective time attributes are retrieved from the row of the input table corresponding to the row counter **714**, **716**. The beginning effective time is subtracted from the ending effective time to create the duration **718**. In one embodiment, the attributes received in step **714** and **716** contain a key to a time dimension table. In such embodiment, the subtraction step **718** first looks up the beginning effective time and ending effective time using the keys received in steps **714**, **716** and the time dimension table, and subtracts the beginning effective time found in the time dimension table from the ending effective time found in the time dimension table. In such embodiment, step **708** also includes the receipt of a time dimension table identifier or name.

The duration identified in step **718** is copied to the record created in step **712**, and this record as modified is appended **720** as the last row of the output table specified in step **708**. If there are more rows in the input table **722**, the row counter is incremented **724** and in the method continues at step **712**. Otherwise, the method terminates **726**.

In another embodiment, state duration is performed as a part or an optional part of temporal normalization described above with reference to Figure **6A**. Referring again to Figure **6A**, in such embodiment, each record created in step **612**

contains a duration attribute. The duration attribute is calculated by subtracting the time corresponding to the beginning effective time attribute of the record from the time corresponding to the ending effective attribute of the record after step **626**, **616** or step **628**.

Referring now to Figure **7B**, a method of performing temporal normalization and state duration is shown according to one embodiment of the present invention. Steps **738**, **740**, **742**, **744**, **746**, **748**, **750**, **752**, **754**, **756**, **758**, **760** and **762** operate as steps **608**, **610**, **612**, **614**, **616**, **618**, **620**, **622**, **624**, **626**, **628**, **630** and **632**, respectively, except as noted below.

Step **738** additionally includes the receipt of a state duration option indicator. Prior to step **758**, if the state duration option indicator indicates state duration is selected **757**, the duration is calculated by the subtraction described above, and stored into a duration attribute created in the record **759**. Otherwise, step **759** is skipped and the method continues at step **758**.

In another embodiment, state duration is always performed, and the duration option received in step **738** is not received as a part of step **738**, step **757** is omitted, and step **759** is performed unconditionally prior to step **758**.

The method described with reference to Figure **7B** may be used in place of the method described with reference to Figures **6A** or **7A** or both.

Transition Detection.

The transition detection query primitive function produces one row in an output table for each transition identified in an input table. The input table may be a fact table, dimension table, or any other table containing some or all of the attributes in the record described above with reference to Figure **2**. A transition is defined as beginning at the beginning effective time attribute of a row for which a second function is true, provided that the row follows a

row for which a first function is true and both such rows have the same time invariant attribute. The two functions are defined to detect the occurrence of a transition. For example, to detect a change in a marital status attributes, the first function may be defined as "marital status = single" and the second function may be defined as "marital status <> single." If a row is identified for which the first function is true and a row follows this row for which a second function is true, if the time invariant attribute is the same for both rows, a transition is detected and the time invariant attribute and the beginning effective time of the second row is copied to a row of the output table.

Referring now to Figure **8A**, a method of performing transition detection is shown according to one embodiment of the present invention. The names of the input and output tables are received, and the two functions are received **808**. A row counter is initialized to a value of '1' **810**. If a row following the row in the input table corresponding to the row counter does not exist **812**, the method terminates **826**. Otherwise, the first function is applied to the row of the input table corresponding to the row counter and if the first function is true **814**, and the row in the input table corresponding to the row counter has a time invariant attribute equal to the time invariant attribute of the following row **816**, and the second function applied to the row in the input table following the row corresponding to the row counter is true **818**, the time invariant attribute, beginning effective time attribute, and optionally, other attributes, of the row following the row of the input table corresponding to the row counter are copied into a new record **820**, which is appended as the last row in the output table **822**. The method continues at step **824**. If any of the tests in step **812**, **814**, **816** or **818** result in "none", "false", "no" or "false",

respectively, steps **820** and **822** are skipped and the method continues at step **824**.

The method shown in Figure **8A** detects transitions occurring only among two adjacent rows of a sorted table. For example, assume a person in the customer table who has a marital status attribute of 'single' in one row, 'engaged' in the next row, and 'married' in the row following. The example functions listed above will detect the transition from single to engaged. However, if the second function is replaced with "marital status = married," no transition will be detected using the method described above with reference to Figure **8A**.

If it desirable to detect transitions which may occur among more than two adjacent rows of the sorted table, another embodiment of the method of the present invention may be used. Referring now to Figure **8B**, a method of performing transition detection is shown according to an alternate embodiment of the present invention.

The names of the input and output tables are received, and the two functions are received **840**. A row counter is initialized to a value of '1' **842**. The time invariant attribute of the row in the input table corresponding to the row counter is selected **846**, for example, by storing it or otherwise referring to it. The first function is applied to the row of the input table corresponding to the row counter. If the first function is false **848**, the method continues at step **862** described below to move to the next row of the input table. Otherwise, the row counter is incremented **850**. If a row does not exist in the input table corresponding to the row counter **852**, the method terminates **866**. Otherwise, the row in the input table corresponding to the row counter is checked **854** to determine whether it has a time invariant attribute that matches the time invariant attribute selected in step **844**. If not, the method continues at step **844**. If

so, the second function is applied to the row in the input table corresponding to the row counter. If the application of the second function to this row produces a false result **856**, the method continues at step **850**. Otherwise, a record is created and the time invariant attribute and the time invariant attribute, beginning effective time, and optionally other attributes, of the row in the input table corresponding to the row counter are copied to this record **858**. The record created and modified in step **858** is appended to the output table **860**, and the row counter is incremented **862**. If a row corresponding to the row counter exists in the input table **864**, the method continues at step **846**. Otherwise, the method terminates **864**.

Example.

A short illustrative example of the method of the present invention will now be provided. Assume a dimensional database such as the database shown above in Figure **1C** with each of the rows in the customer dimension table arranged as described with respect to the record of Figure **2** is accessible to the user. The customer dimension table has rows containing the time invariant attribute, TIA, beginning effective time attribute, BTA, ending effective time attribute, ETA and surrogate attribute, Surrogate_Key. Each row of the customer dimension table has a "marital status" attribute, with possible values of 'single' or 'married'.

The fact table contains a "units" attribute containing the number of units in the order, and a customer_key containing a key to the customer table.

The time dimension table contains two columns for each row: a "Date" column containing a date, and a "Time_Key" containing the key to the row.

Assume a user of the database wishes to know the number of units sold to customers who have been married for two or more years at any time. The user may first perform a

conventional selection from the customer table, selecting records from the customer table where the marital status attribute is equal to 'Married'. Assume the selection is performed so that the output table of the selection is the customer table. For example, using the conventional SQL query:

SELECT TIA, BTA, ETA FROM customer_dimension WHERE marital_status = 'Married' (Statement 0)

The user may sort the resulting table as described above and perform a temporal normalization on the resulting table as described above with reference to Figures **6A, 6B** or **6C**, for example using the commands set forth below:

temporal_normalization with
input table := customer_dimension
output table := normalized_customer_dimension having TIA, BTA and ETA attributes
selected attributes := marital status (Statement 1)

The above command performs temporal normalization as described with reference to Figure **6A** on the input table, copying the time invariant attribute, beginning effective time attribute and ending effective time attribute from the customer dimension table into the normalized customer dimension output table, and consolidating rows as described above.

The user next will create (i.e. define) a table in the database having a single column of the same type as the time invariant attributes of the customer dimension table. Assume the user creates this table and calls it "married_customers_table" with the attribute Married_TIA.

The user will then select the time invariant attribute from rows meeting the criteria of having been married for more than two years, and place them into the married_customers table created above. For example, for each row of the normalized_customer_dimension table:

```
SELECT Date AS StartDate FROM time_dimension WHERE
Time_Key = BTA
SELECT Date AS StopDate FROM time_dimension WHERE
Time_Key = ETA
IF StopDate - StartDate > 2 years
    IF there is not a row in married_customers_table
where Married_TIA = TIA
        Append a new row to married_customers_table
having Married_TIA = TIA
        ENDIF
    ENDIF                                (Statement 2)
```

The lookup of the dates in the time dimension table is performed to accommodate the variance in number of days due to leap years. The "IF" statement is performed to eliminate duplicate entries for people who are married for more than two years, then not married, and then married again for more than two years, although as used below, the elimination of the duplicate entries will be seen as not necessary.

Next, the fact table and customer dimension table are joined to produce the result of the temporal query, for example,

```
SELECT Units
    FROM fact, customer_dimension
    WHERE fact.customer_key =
customer_dimension.Surrogate_Key
        AND customer_dimension.TIA IN
            (Select
married_customers_table.Married_TIA FROM
married_customers_table)                (Statement 3)
```

Referring again to Figure **3A**, statements 0, 1, 2, and 3 may be received as a part of a temporal query, and the temporal query is performed as described above to produce the result.

Intermediate Tables and Key Set Generation.

It can be helpful to use the output of a temporal query primitive function to generate a table containing keys to one of the dimension or fact tables that match one or more criteria of interest to a user. Such table is known as a key set table. In the example listed above, the WHERE clause in Statement 3 produces a key set table to join with the fact table.

However, if desired, the key set table may be placed into a separate table for use in other statements. This is performed by joining a dimension table with a result of a temporal query primitive function or an intermediate table produced from the result of a temporal query primitive function using a conventional selection, projection or join command. In the example above, the married_customers_table built using statement 2 is an intermediate table produced from the result of the temporal normalization performed in statement 1, and the second SELECT in Statement 3 uses this intermediate table to build another intermediate table.

A key set table could have been produced using the conventional query:

```
SELECT Surrogate_Key
     FROM customer_dimension
     WHERE customer_dimension.TIA IN
          (SELECT married_customers_table.TIA FROM
married_customers_table)
```

Apparatus.

Referring now to Figure 9, a system for selecting records from a fact table of a dimensional database by performing temporal query primitive functions and one or more joins on the tables of a dimensional database is shown according to one embodiment of the present invention. Fact table storage **940** stores the fact table of the dimensional database. Time dimension table storage **946** stores the time dimension table of the dimensional database. Dimension table

storage **970** holds one or more remaining dimension tables of the dimensional database. Temporal selection module **916**, temporal projection module **914**, temporal normalization module **918**, state duration module **920**, transition detection module **922** perform the corresponding temporal query primitive functions described above.

The system of Figure **9** performs the functions described above with reference to Figure **3A**. Query module **960** receives at input/output **908** temporal queries in a query language such as SQL or a modified SQL format or non-SQL format such as is described in the OLE DB for OLAP specification at http://www.microsoft.com/data/oledb/olap, the MD-API described at http://www.olapcouncil.org, ODMG 2.0 available at http://www.odmg.org or any other query language that implements conventional selection, conventional projection and conventional joins. Query module **960** parses the query, API or other command received at input/output **908**, and directs sorter module **912**, temporal projection module **914**, temporal selection module **916**, temporal normalization module **918**, state duration module **920**, transition detection module **922** to perform their temporal query primitive functions required by the query received at input/output **908**. Query module **960** also directs join module **930** and select/project module **932** to perform functions indicated by the temporal query received at input/output **908**.

Query module **960** sends the table names and other information used to perform any of the functions described above and an identifier to identify the module **916, 912, 914, 918, 920, 922, 930, 932** that will perform the query primitive or other function described herein. In response to the receipt of the corresponding identifier of the module **916, 912, 914, 918, 920, 922, 930, 932**,the module **916, 912, 914, 918, 920, 922, 930, 932** locates the table or tables having the name received from query module **960**, performs the

function described below using the information received from query module **960** and stores the result in storage areas **940, 942, 946, 950, 970, 972, 974** and **976**, which may be any storage device such as conventional memory or disk, as described herein. Query module then provides the result of the query at input/output **908**.

If desired, sorter module **970** sorts the tables in dimension table storage **970** or fact table storage **940**. Sorter module **912** receives from query module **960** at input **911** the name of the table to be sorted, the list of attributes or keys, in sort priority order, on which to sort the table, and whether each attribute is to be sorted in ascending or descending order. Sorter module **912** sorts the table stored in dimension table storage **970** or fact table storage **940**. As described above, the sorting may be performed by physically sorting the records in the table, or logically sorted by creating in the same storage area as the table an index for the table that relates a numerical sort order to each row of the table.

Temporal projection module **914** receives from query module **960** at input **913** the names of the input and output tables and other inputs to be used for a temporal projection as described above with reference to Figures **4A, 4B1, 4B2, 4B3**, or **4C**. Temporal projection module **914** reads the input table specified at input **913** in dimension table storage **970** or fact table storage **940** and creates in projection table storage **950** a projection table having the name of the output table received at input **913**.

In one embodiment, temporal projection module **914** also receives at input **913** the names of the attributes corresponding to the beginning effective time, ending effective time or status. Temporal projection module **914** uses the names of these attributes to identify the proper attributes, and to determine which of the methods of Figures

4A, 4B1, 4B2, 4B3 and 4C it will use to perform the temporal projection as described above with reference to Figure 4D.

Temporal selection module 916 receives from query module 960 at input 915 the function, input table name, the temporal projection table name and other inputs as described above with reference to Figure 5. Temporal projection module 914 reads the temporal projection table having the name received at input 916 in projection table storage 950 and performs the temporal projection described above with reference to Figure 5 by selecting the records in the input table stored in dimension table storage 970 or fact table storage 940. If no output table name is specified, temporal selection module 916 selects the rows of the input table by adjusting the pointers for the table as described above. Temporal selection module 916 may optionally receive at input 913 the name of an output table. If the name of an output table is received, instead of selecting records in the input table, temporal selection module creates the output table in dimension table storage 970 by creating separate arrays or dynamic arrays of pointers having the name of the output table and pointing to the records of the input table in dimension table storage 970 that are selected in Figure 4A, 4B1, 4B2, 4B3 or 4C.

In one embodiment, if a query received by query module 960 at input/output 908 contains a command requiring temporal selection, and if no projection table exists for the table to be temporally selected, query module 960 first instructs temporal projection module 914 to create such a projection table in projection table storage 950. Query module 960 then directs temporal selection module 916 to utilize the created projection table in projection table storage 950.

Temporal normalization module 918 receives from query module 960 at input 917 the name of the input and output tables and the selected attribute identifiers and other inputs and performs the temporal normalization described

above with reference to Figures 6A, 7B, 6B or 6C. In one embodiment, temporal normalization module 918 also receives at input 917 the attribute names corresponding to the beginning and ending effective time attributes or the status attribute required to implement the temporal normalization query primitive function. Temporal normalization module 918 selects the method used to perform the temporal normalization query primitive from those of Figures 6A, 6B or 6C depending on which attribute names are received as described above with reference to Figure 6D.

Temporal normalization module 918 reads the input table specified from dimension table storage 970 or fact table storage 940. Temporal normalization module uses the method described above with reference to Figures 6A, 6B or 6C to create a table having the output name specified at input 917 in normalized dimension table storage 972 if the input table was located in dimension table storage 970 or normalized dimension table storage 972. Temporal normalization module 918 creates a table having the output name specified at input 917 into normalized fact table storage 942 if the input table was located in fact table storage 940 or normalized fact table storage 942.

State duration module 920 receives from query module 960 at input 919 names of the input and output tables other inputs, and performs the state duration function described above with reference to Figures 7A or 7B on the table described at input 919. State duration module 920 reads the input table from dimension table storage 970, normalized dimension table storage 972, fact table storage 940 or normalized fact table storage 942. State duration module 920 creates a state duration table as described above with reference to Figure 7A and stores the state duration table in state duration table storage 974 with the name as received at input 919. Alternatively, the state duration is stored in

the normalized table stored in normalized dimension table storage **972** or normalized fact table storage **942** using the method described above with reference to Figure **7B**.

Transition detection module **922** receives from query module **960** the names of input and output tables and two functions and other inputs at input **921**. Transition detection module **922** reads the input table from dimension table storage **970**, state duration table storage **974**, normalized dimension table storage **972**, projection table storage **950**, normalized fact table storage **942** or fact table storage **940**, and performs the transition detection function described above with reference to Figure **8A**, **8B** or both. Transition detection module creates the output table having the name specified at input **921** in transition table storage **976**.

In one embodiment, any of the temporal query primitive modules **914, 916, 918, 920, 922** can obtain input tables from any storage **940, 942, 946, 950, 970, 972, 974, 976**.

Select/project module **932** receives from query module **960** conventional selection and projection commands, including the name of the input tables stored in any storage area **940, 942, 946, 950, 970, 972, 974, 976**, attributes and the name of the output table. Select/project module **932** builds the output table by performing the non-temporal selections or projections described with reference to step **310** of Figure **3A**, and may be used in building intermediate tables described in step **321** of Figure **3A**. Output tables are placed into projection table storage **950** if the function is a projection, or the storage area **940, 942, 946, 950, 970, 972, 974, 976** from which the input table was retrieved.

Join module **930** receives at input **929** the name of any two tables in fact table storage **940** dimension table storage **970**, state duration table storage **974**, transition table storage **976**, normalized dimension table storage **972**,

projection table storage **950**, normalized fact table storage **942**, or time dimension table storage **946**. Join module **930** receives at input **929** the names and attributes or keys of the tables to join. Join module **930** performs the join described above with reference to Figure **3B**, selecting the records as described above. In one embodiment, join module **930** provides conventional join capabilities, such as those found in the commercially available Oracle or Access database products.

In one embodiment, join module **930** receives at input **929** the name of an output file. Instead of selecting records from the input table, join module **930** creates in fact table storage **940** an output table having the name received at input **929** with records selected from the fact table in fact table storage **940**.

As used herein, the phrase "one selected from", and the like does not necessarily imply a database selection described above.

What is claimed is:

1. A method of selecting records from at least one first table of at least one dimensional database, the method comprising:

fashioning at least one second table responsive to a time attribute of at least one of the at least one first tables of the at least one dimensional database; and

responsive to at least one of the at least one second table, picking at least one row from one selected from a first table of the at least one dimensional database and a table produced responsive to at least a portion of at least one first table of the at least one dimensional database.

2. The method of claim 1 wherein the picking step comprises joining at least one row from the one selected from the first table of the at least one dimensional database with the table produced from a first table of the at least one dimensional database.

3. The method of claim 1 wherein the picking step comprises selecting at least one row from the one selected from the first table of the at least one dimensional database with the table produced responsive to a first table of the at least one dimensional database.

4. The method of claim 1 comprising the additional step of picking, independently of the second table, at least one row from a table selected from at least one of the first tables of the at least one dimensional database and a table produced responsive to at least one of the first tables of the at least one dimensional database.

5. The method of claim 4 additionally comprising the step of fashioning at least one intermediate table from at least one of the at least one second table; and

wherein the picking step responsive to the portion of the at least one second table is additionally responsive to at least a portion of at least one intermediate table.

6. The method of claim 1, wherein the fashioning step comprises:

receiving an identifier of a first input table comprising at least a portion of one of the first tables, the first input table comprising a plurality of rows, at least one row comprising at least one selected from an identifier of a beginning effective time and an identifier of an ending effective time; and

fashioning a third table comprising at least one row comprising at least one selected from at least one identifier corresponding to at least one of the identifiers of the beginning effective time from at least one row of the first input table and at least one identifier corresponding to at least one of the identifiers of the ending effective time from at least one row of the first input table.

7. The method of claim 6, wherein the fashioning step additionally comprises:

receiving an identifier of a second input table, the second input table comprising at least a portion of one selected from at least one of the first tables and a table fashioned responsive to at least one of the first tables, the second input table comprising a plurality of rows comprising at least one selected from at least one attribute and at least one key;

receiving a function having a result having a first state and a second state; and

for each of a plurality of the rows in the third table:

picking the row in the third table;

applying the function to at least one of the identifiers in the row picked; and

responsive to the result of the function applied to the row having the first state, picking from the second table at least one selected from the at least one attribute and at least one key, of a row of the second input table corresponding to the row of the third table picked.

8. The method of claim 1, wherein the fashioning step comprises:

receiving an identifier of an input table, the input table comprising at least a portion of one selected from at least one of the first tables and a table fashioned responsive to at least one of the first tables, the input table comprising a plurality of rows comprising at least one effective time attribute, at least one first attribute having a value and at least one second attribute, each second attribute having a value;

comparing the value of the first attribute in one row of the input table with the value of the first attribute of at least one different row of the input table;

comparing each value of at least one of the second attributes of the one row of the input table with each value of at least one of the second attributes of the at least one different row of the input table; and

for each of a plurality of sets of at least two rows in the input table having a same value of the first attribute and a same value of at least one of the second attributes, fashioning one row of the second table for each said set of rows, each of said row fashioned in the second table comprising:

the value of the first attribute; and

the value of at least one of the second attributes.

9. The method of claim 1, wherein the fashioning step comprises:

receiving an identifier of an input table, the input table having a plurality of rows comprising a first effective time identifier and having a second effective time; and

fashioning the second table with a plurality of rows, a plurality of the rows of the input table containing a value corresponding to a value of the identifier of the first effective time of a row of the input table subtracted from a value of the second effective time of the row of the input table.

10. The method of claim 9 wherein the value of the second time of at least one of the rows of the input table is identified using a different row of the input table.

11. The method of claim 1, wherein the fashioning step comprises:

receiving an identifier of an input table comprising rows comprising information comprising at least one identifier of a time and at least one attribute;

receiving a first function having a result having a first state and a second state and a second function having a result having a first state and a second state; and

for each of a plurality of sets of a plurality of rows of the input table, each set of rows comprising a first row and a second row:

applying the first function to the information of the first row;

applying the second function to the information of the second row;

comparing at least one of the attributes of the first row with at least one of the attributes of the second row; and

responsive to the result of the applied first function having the first state, the result of the applied second function having a first state, and the at least one attribute compared in the first row matching the at least one attribute compared in the second row, fashioning the second table to have a row comprising at least a portion of the information in one selected from the first row and the second row, the portion of information comprising at least one of the identifiers of a time.

12. The method of claim 11, wherein the time identifiers of each of the rows in each of the sets indicate the rows in each set are contiguous in time.

13. The method of claim 11, wherein each set of rows comprises two rows.

14. The method of claim 13 wherein the time identifiers of the two rows of each set indicate the two rows in each set are contiguous in time.

15. The method of claim 11, comprising the additional step of, for each of a plurality of rows of the second table:

subtracting a first identifier of a time from a second identifier of a time to produce a subtraction result; and

storing the subtraction result in said row of the second table.

16. A computer program product comprising a computer useable medium having computer readable program code embodied therein for selecting records from at least one first table

of at least one dimensional database, the computer program product comprising:

computer readable program code devices configured to cause a computer to fashion at least one second table responsive to a time attribute of at least one of the at least one first tables of the at least one dimensional database; and

computer readable program code devices configured to cause a computer to pick, responsive to at least one of the at least one second table, at least one row from one selected from a first table of the at least one dimensional database and a table produced responsive to at least a portion of at least one first table of the at least one dimensional database.

17. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to pick step comprise computer readable program code devices configured to cause a computer to join at least one row from the one selected from the first table of the at least one dimensional database with the table produced from a first table of the at least one dimensional database.

18. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to pick comprise computer readable program code devices configured to cause a computer to select at least one row from the one selected from the first table of the at least one dimensional database with the table produced responsive to a first table of the at least one dimensional database.

19. The computer program product of claim 16, additionally comprising computer readable program code

devices configured to cause a computer to pick, independently of the second table, at least one row from a table selected from at least one of the first tables of the at least one dimensional database and a table produced responsive to at least one of the first tables of the at least one dimensional database.

20. The computer program product of claim 19 additionally comprising computer readable program code devices configured to cause a computer to fashion at least one intermediate table from at least one of the at least one second table; and

wherein the computer readable program code devices configured to cause a computer to pick responsive to the portion of the at least one second table are additionally responsive to at least a portion of at least one intermediate table.

21. The computer program product of claim 16, wherein the computer readable program code devices configured to cause a computer to fashioning comprise:

computer readable program code devices configured to cause a computer to receive an identifier of a first input table comprising at least a portion of one of the first tables, the first input table comprising a plurality of rows, at least one row comprising at least one selected from an identifier of a beginning effective time and an identifier of an ending effective time; and

computer readable program code devices configured to cause a computer to fashion a third table comprising at least one row comprising at least one selected from at least one identifier corresponding to at least one of the identifiers of the beginning effective time from at least one row of the first input table and at least one identifier corresponding

to at least one of the identifiers of the ending effective time from at least one row of the first input table.

22. The computer program product of claim 21, wherein the computer readable program code devices configured to cause a computer to fashion additionally comprise:

computer readable program code devices configured to cause a computer to receive an identifier of a second input table, the second input table comprising at least a portion of one selected from at least one of the first tables and a table fashioned responsive to at least one of the first tables, the second input table comprising a plurality of rows comprising at least one selected from at least one attribute and at least one key;

computer readable program code devices configured to cause a computer to receive a function having a result having a first state and a second state; and

for each of a plurality of the rows in the third table:

computer readable program code devices configured to cause a computer to pick a row in the third table;

computer readable program code devices configured to cause a computer to apply the function to at least one of the identifiers in the row picked; and

computer readable program code devices configured to cause a computer to pick from the second table at least one selected from the at least one attribute and at least one key, of a row of the second input table corresponding to the row of the third table picked, responsive to the result of the function applied to the row having the first state.

23. The computer program product of claim 16, wherein the computer readable program code devices configured to cause a computer to fashion comprise:

computer readable program code devices configured to cause a computer to receive an identifier of an input table, the input table comprising at least a portion of one selected from at least one of the first tables and a table fashioned responsive to at least one of the first tables, the input table comprising a plurality of rows comprising at least one effective time attribute, at least one first attribute having a value and at least one second attribute, each second attribute having a value;

computer readable program code devices configured to cause a computer to compare the value of the first attribute in one row of the input table with the value of the first attribute of at least one different row of the input table;

computer readable program code devices configured to cause a computer to compare each value of at least one of the second attributes of the one row of the input table with each value of at least one of the second attributes of the at least one different row of the input table; and

computer readable program code devices configured to cause a computer to, for each of a plurality of sets of at least two rows in the input table having a same value of the first attribute and a same value of at least one of the second attributes, fashion one row of the second table for each said set of rows, each of said row fashioned in the second table comprising:

the value of the first attribute; and

the value of at least one of the second attributes.

24. The computer program product of claim 16, wherein the computer readable program code devices configured to cause a computer to fashion comprise:

computer readable program code devices configured to cause a computer to receive an identifier of an input table, the input table having a plurality of rows comprising a first effective time identifier and having a second effective time; and

computer readable program code devices configured to cause a computer to fashion the second table with a plurality of rows, a plurality of the rows of the input table containing a value corresponding to a value of the identifier of the first effective time of a row of the input table subtracted from a value of the second effective time of the row of the input table.

25. The computer program product of claim 24 wherein the value of the second time of at least one of the rows of the input table is identified using a different row of the input table.

26. The computer program product of claim 16, wherein the computer readable program code devices configured to cause a computer to fashion comprises:

computer readable program code devices configured to cause a computer to receive an identifier of an input table comprising rows comprising information comprising at least one identifier of a time and at least one attribute;

computer readable program code devices configured to cause a computer to receive a first function having a result having a first state and a second state and a second function having a result having a first state and a second state; and

computer readable program code devices configured to cause a computer to, for each of a plurality of sets of a plurality of rows of the input table, each set of rows comprising a first row and a second row:

apply the first function to the information of the first row;

apply the second function to the information of the second row;

compare at least one of the attributes of the first row with at least one of the attributes of the second row; and

responsive to the result of the applied first function having the first state, the result of the applied second function having a first state, and the at least one attribute compared in the first row matching the at least one attribute compared in the second row, fashion the second table to have a row comprising at least a portion of the information in one selected from the first row and the second row, the portion of information comprising at least one of the identifiers of a time.

27. The computer program product of claim 26, wherein the time identifiers of each of the rows in each of the sets indicate the rows in each set are contiguous in time.

28. The computer program product of claim 26, wherein each set of rows comprises two rows.

29. The computer program product of claim 28 wherein the time identifiers of the two rows of each set indicate the two rows in each set are contiguous in time.

30. The computer program product of claim 26, additionally comprising computer readable program code devices configured to cause a computer to, for each of a plurality of rows of the second table:

subtract a first identifier of a time from a second identifier of a time to produce a subtraction result; and

store the subtraction result in said row of the second table.

31. A system for selecting rows from a table, comprising:

at least one storage device having an input/output for receiving and providing a plurality of first tables in at least one dimensional database and at least one second table;

a query module having an input/output operatively coupled to receive a query and at least one output for providing at least one identifier of at least one table stored in the storage device responsive to the query received at the query module input/output, the query module for parsing the query, transmitting at the query module output identifiers of said tables responsive to the query and providing the result of the query at the query module input/output;

a set of at least one temporal query primitive module, comprising an input coupled to the query module output for receiving an identifier of at least one table, an input/output coupled to each of the input/outputs of at least one of the storage devices, each temporal query primitive module in the set for fashioning in the storage device an output table responsive to at least one time attribute of at least a portion of the table having at least one of the identifiers received at the temporal query primitive module input; and

a join module having an input coupled to the query module output for receiving an identifier of a plurality of tables stored in the storage device, the join module for fashioning in at least one of the storage devices a second table by picking at least one row from at least a first one of the tables having an identifier received at the join

module input responsive to at least a portion of at least one second table having an identifier received at the join module input, wherein at least one first table having an identifier received at the join module input and at least one second table having an identifier received at the join module input comprises an output table.

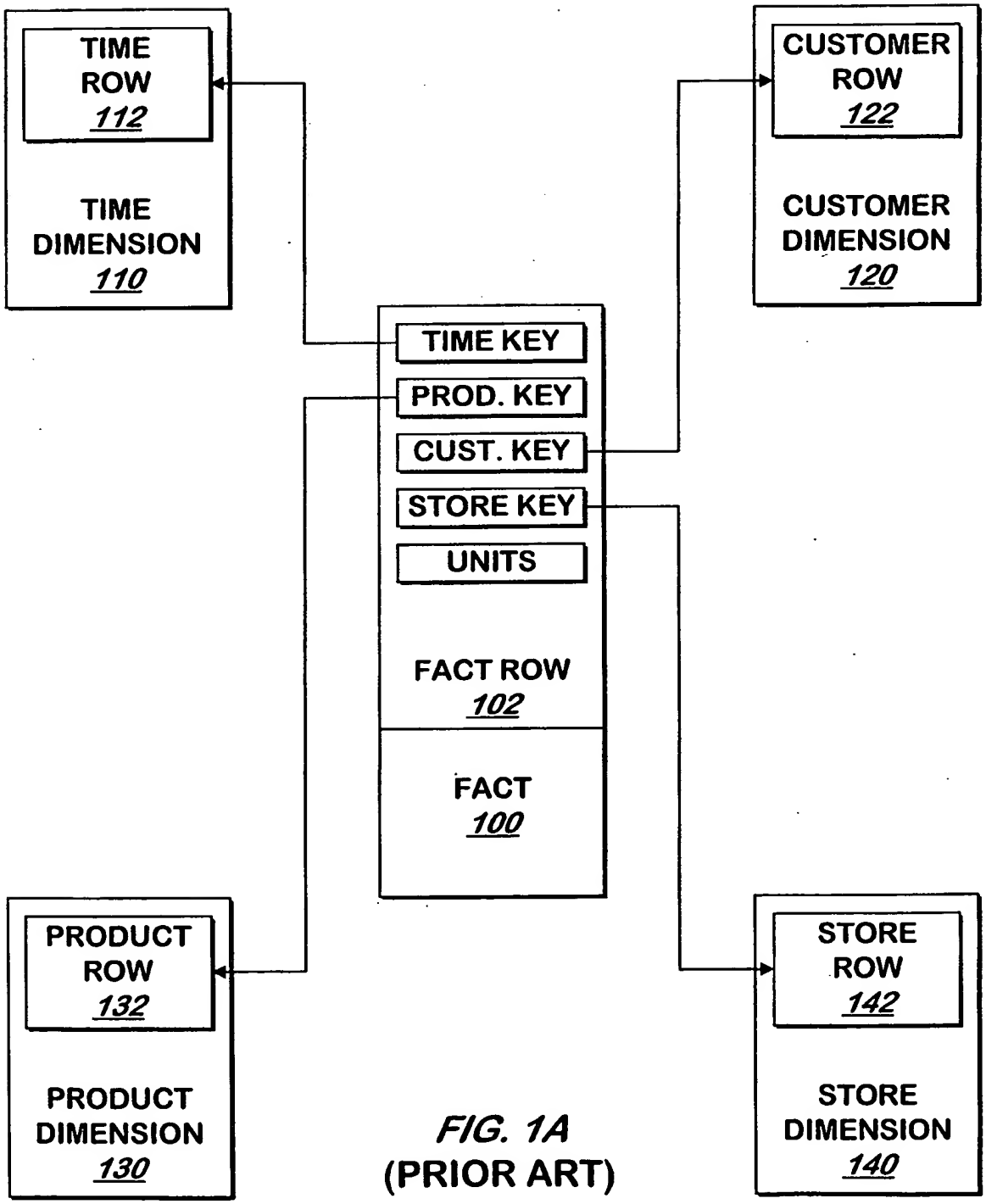


FIG. 1A
(PRIOR ART)

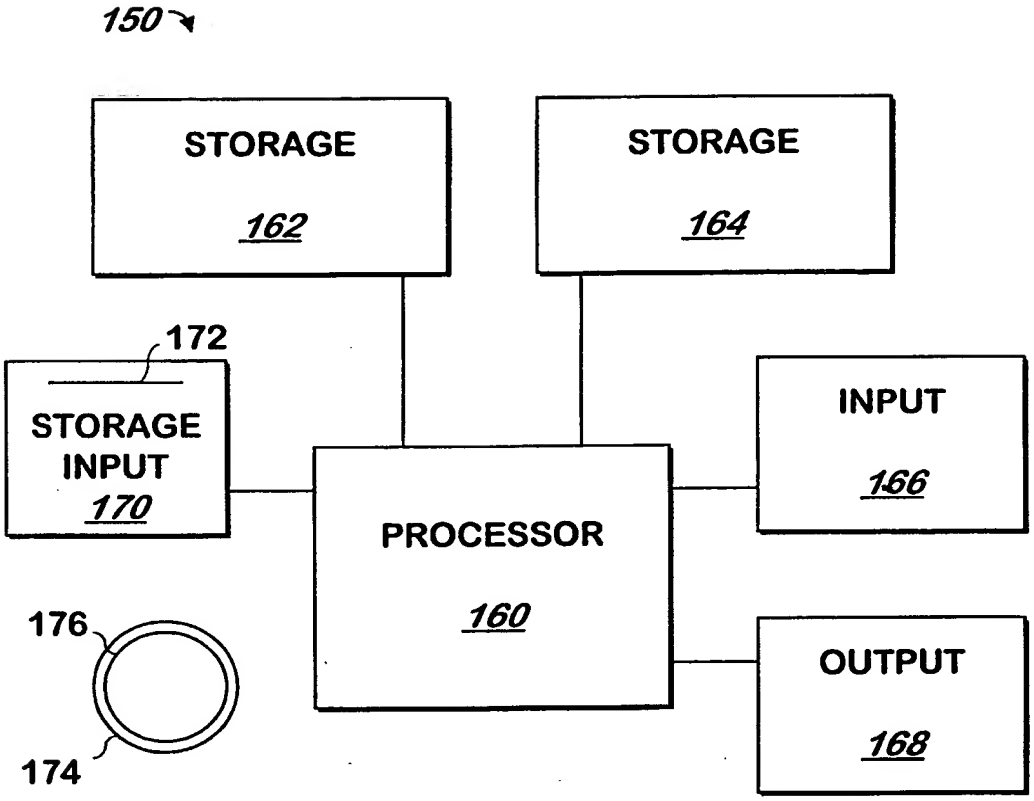


FIG. 1B
(PRIOR ART)

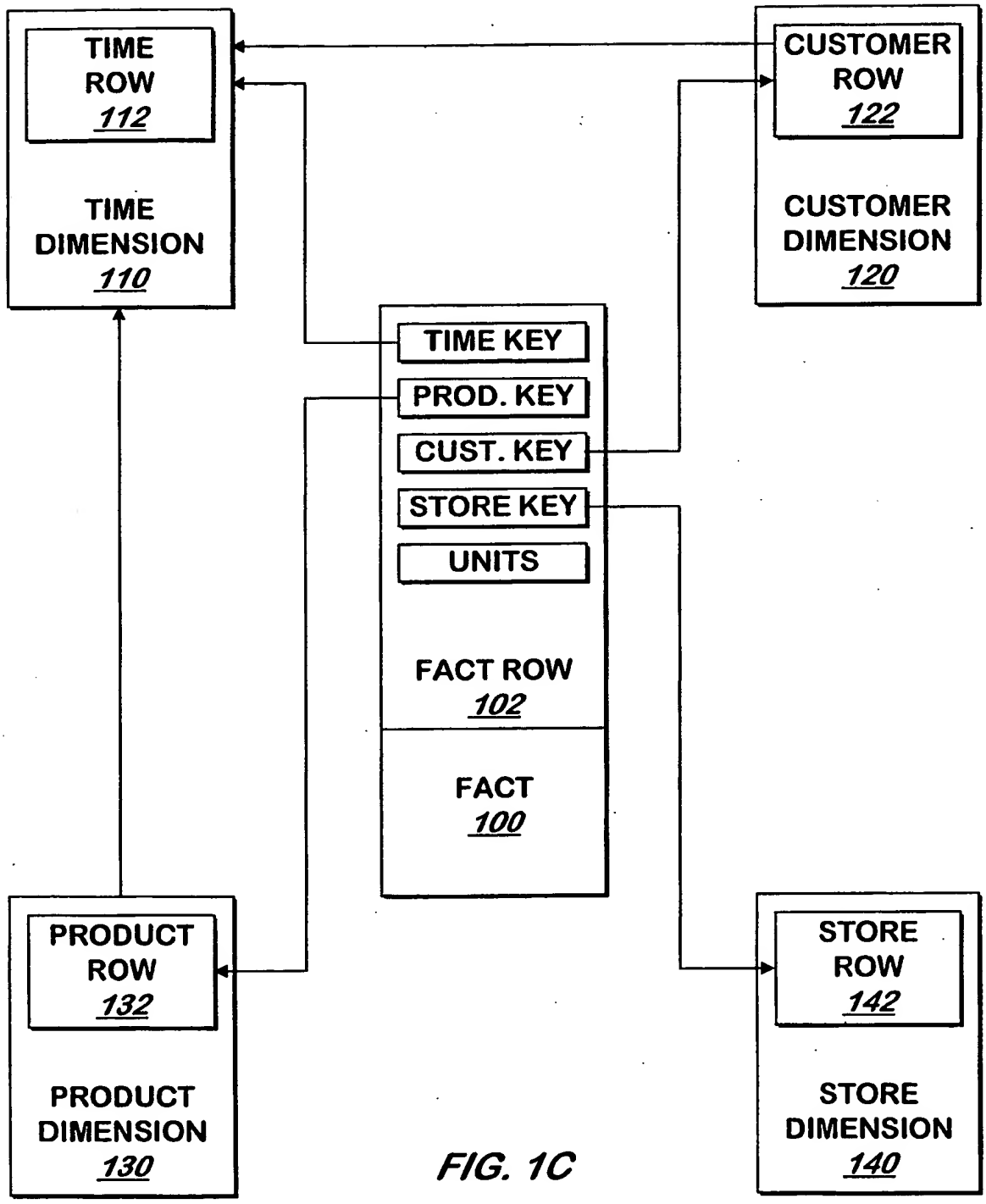

TIME ROW 112
TIME DIMENSION 110
CUSTOMER ROW 122
CUSTOMER DIMENSION 120
TIME KEY
PROD. KEY
CUST. KEY
STORE KEY
UNITS
FACT ROW 102
FACT 100
PRODUCT ROW 132
PRODUCT DIMENSION 130
STORE ROW 142
STORE DIMENSION 140

FIG. 1C

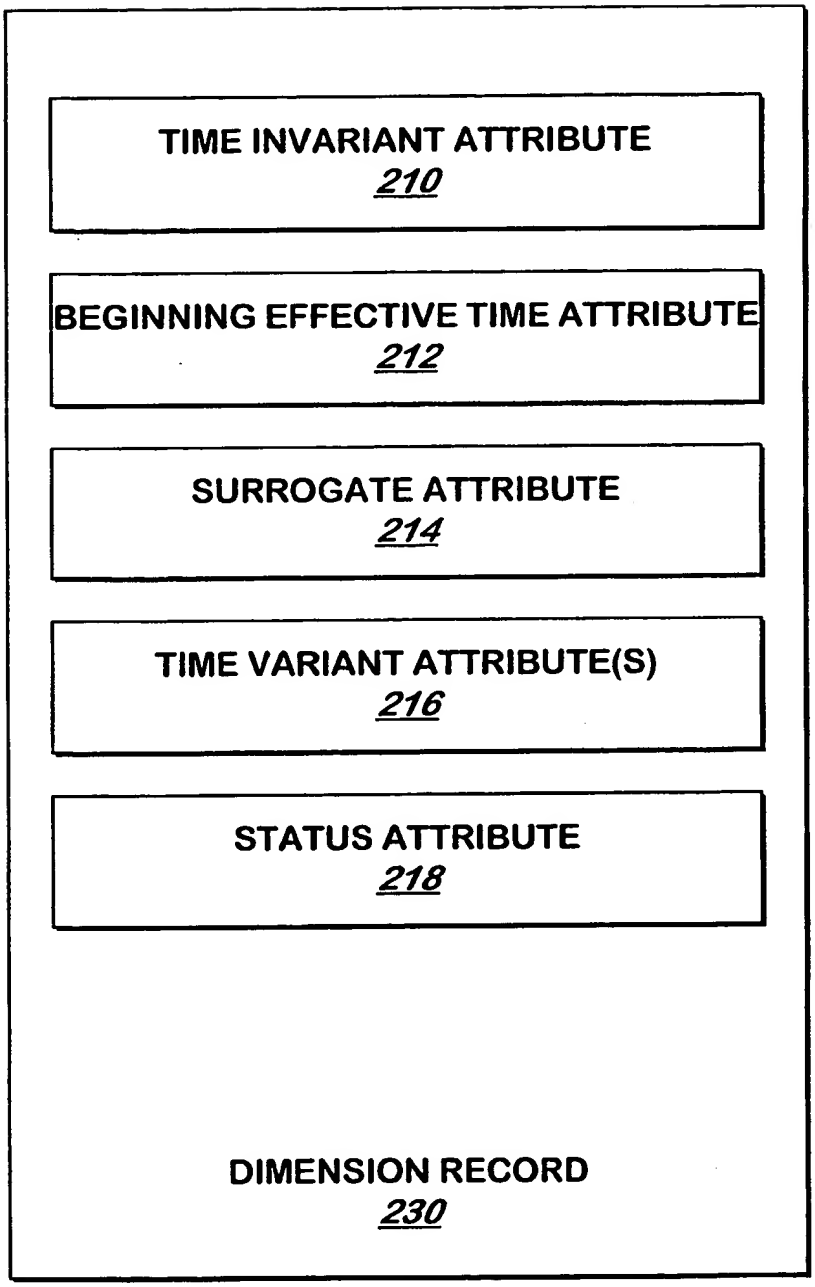

TIME INVARIANT ATTRIBUTE
210
BEGINNING EFFECTIVE TIME ATTRIBUTE
212
SURROGATE ATTRIBUTE
214
TIME VARIANT ATTRIBUTE(S)
216
STATUS ATTRIBUTE
218
DIMENSION RECORD
230

FIG. 2

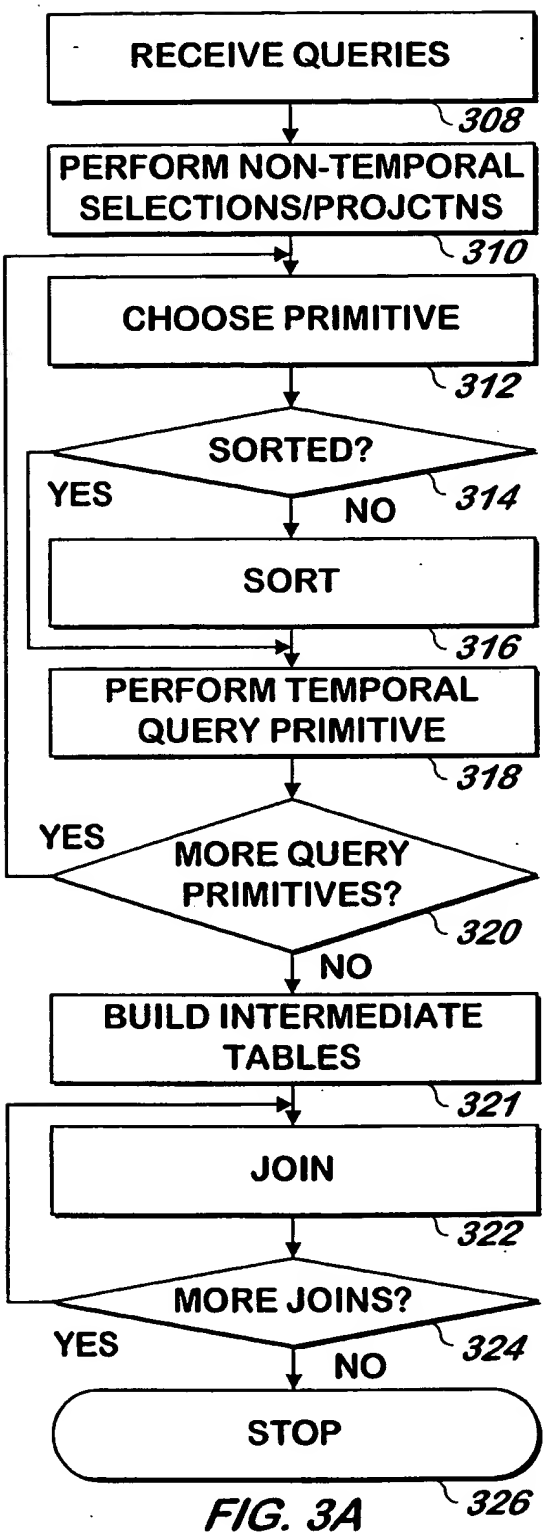

RECEIVE QUERIES
308
PERFORM NON-TEMPORAL SELECTIONS/PROJCTNS
310
CHOOSE PRIMITIVE
312
SORTED?
YES
NO
314
SORT
316
PERFORM TEMPORAL QUERY PRIMITIVE
318
YES
MORE QUERY PRIMITIVES?
320
NO
BUILD INTERMEDIATE TABLES
321
JOIN
322
MORE JOINS?
YES
NO
324
STOP
326

FIG. 3A

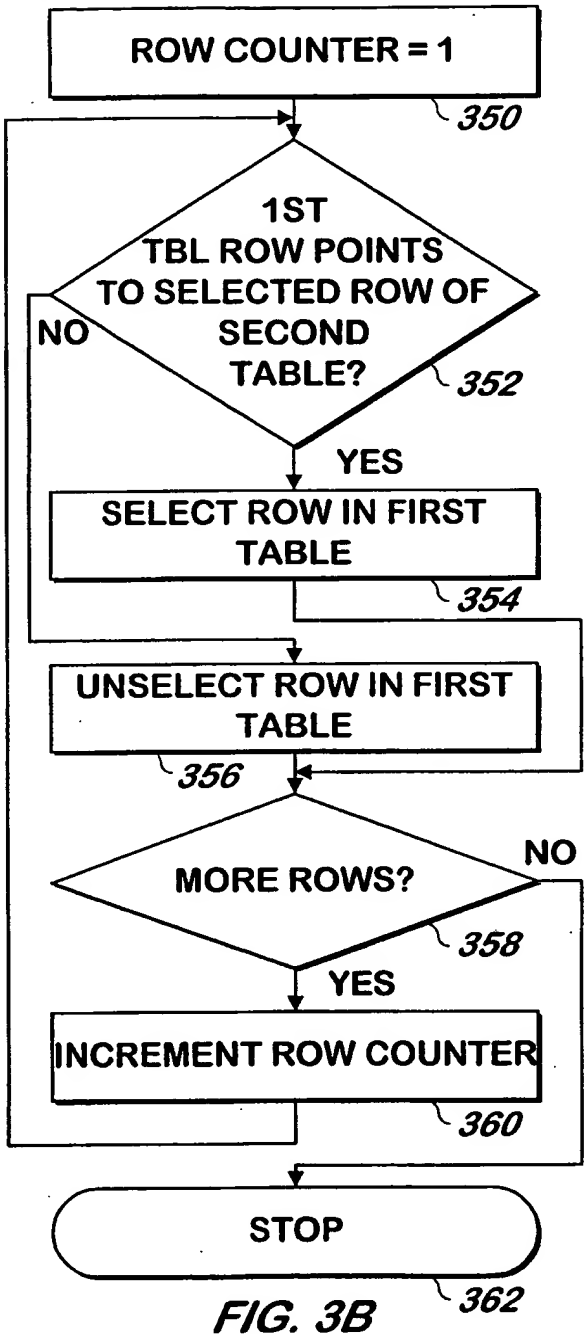

ROW COUNTER = 1
350
1ST TBL ROW POINTS TO SELECTED ROW OF SECOND TABLE?
NO
352
YES
SELECT ROW IN FIRST TABLE
354
UNSELECT ROW IN FIRST TABLE
356
MORE ROWS?
NO
358
YES
INCREMENT ROW COUNTER
360
STOP
362

FIG. 3B

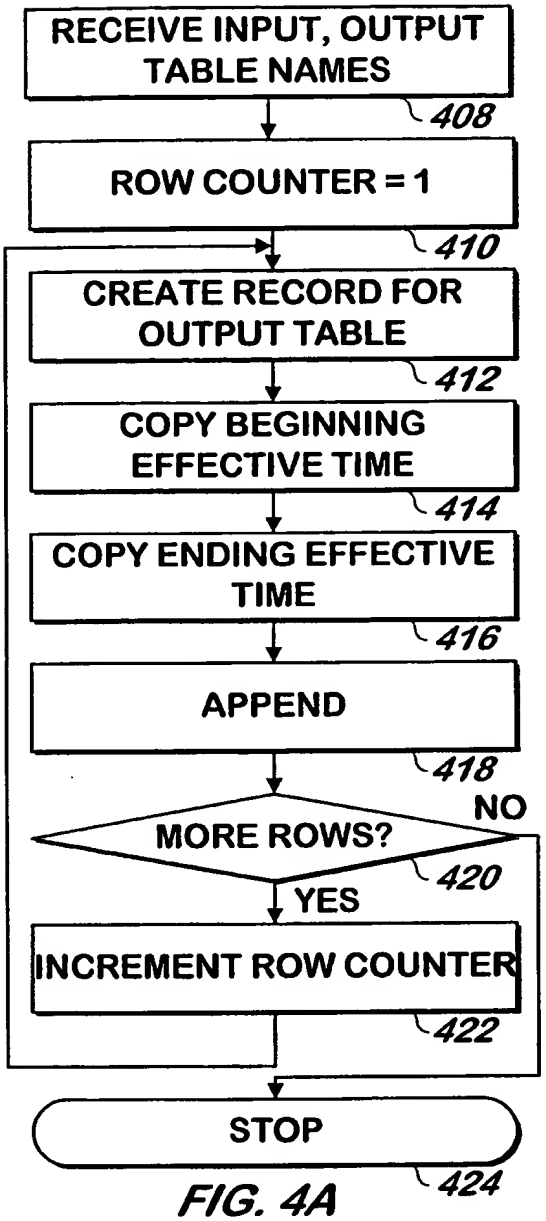


FIG. 4A

FIG. 4B1

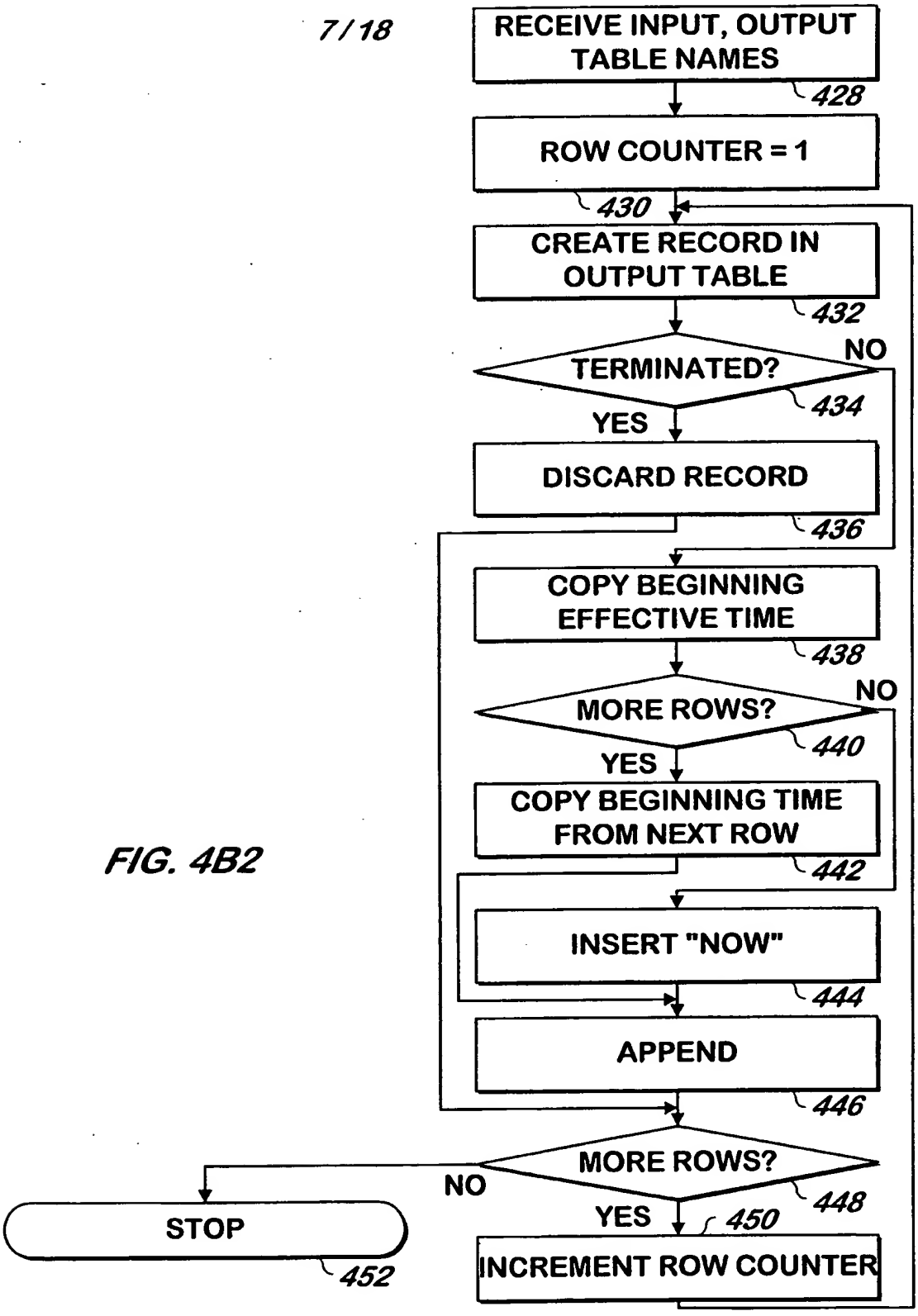

RECEIVE INPUT, OUTPUT TABLE NAMES
428
ROW COUNTER = 1
430
CREATE RECORD IN OUTPUT TABLE
432
TERMINATED?
NO
434
YES
DISCARD RECORD
436
COPY BEGINNING EFFECTIVE TIME
438
MORE ROWS?
NO
440
YES
COPY BEGINNING TIME FROM NEXT ROW
442
INSERT "NOW"
444
APPEND
446
MORE ROWS?
NO
448
YES
450
INCREMENT ROW COUNTER
STOP
452

FIG. 4B2

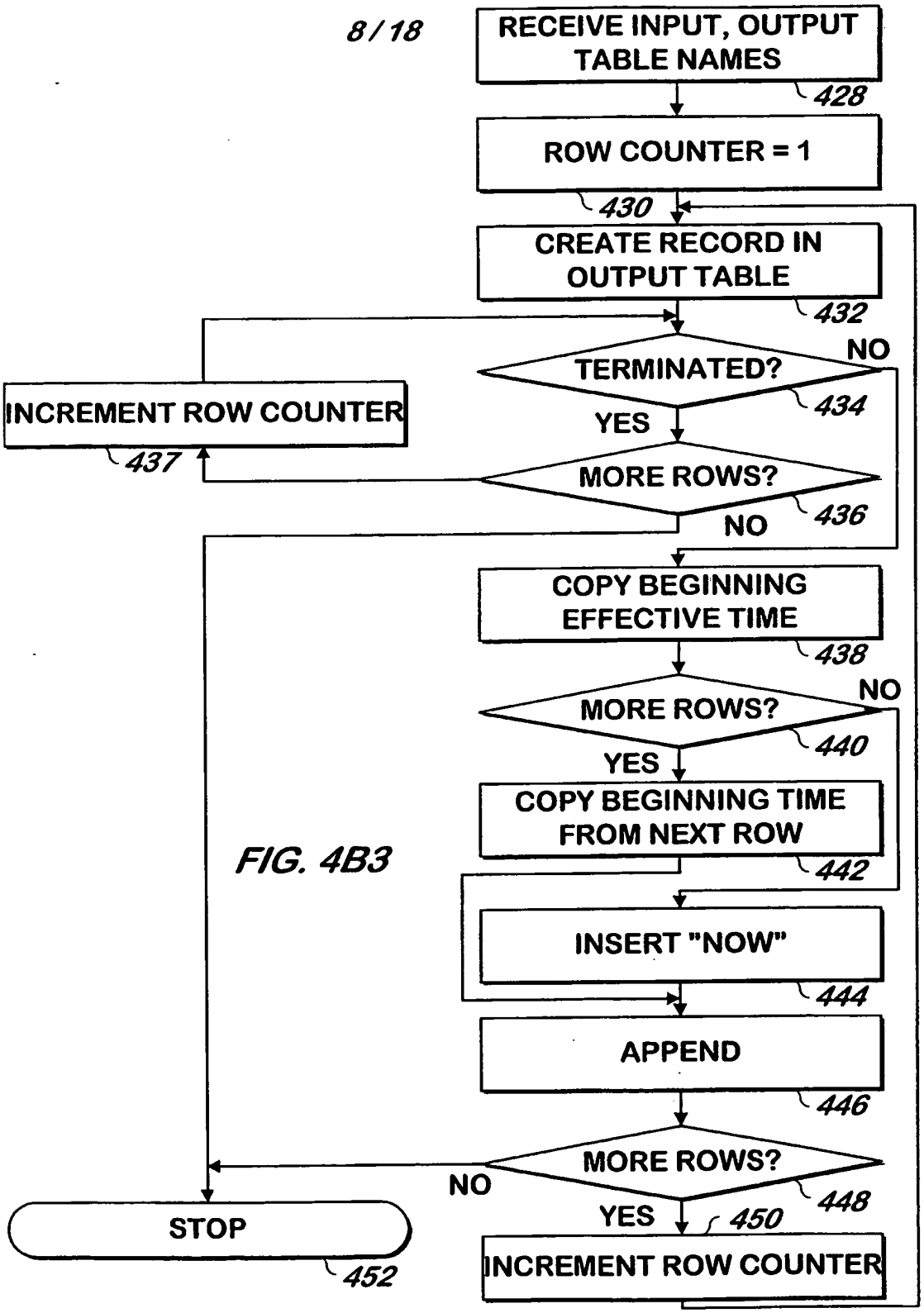

RECEIVE INPUT, OUTPUT TABLE NAMES
428
ROW COUNTER = 1
430
CREATE RECORD IN OUTPUT TABLE
432
TERMINATED?
NO
434
YES
INCREMENT ROW COUNTER
437
MORE ROWS?
436
NO
COPY BEGINNING EFFECTIVE TIME
438
MORE ROWS?
NO
440
YES
COPY BEGINNING TIME FROM NEXT ROW
442
INSERT "NOW"
444
APPEND
446
MORE ROWS?
NO
448
YES
450
INCREMENT ROW COUNTER
STOP
452

FIG. 4B3

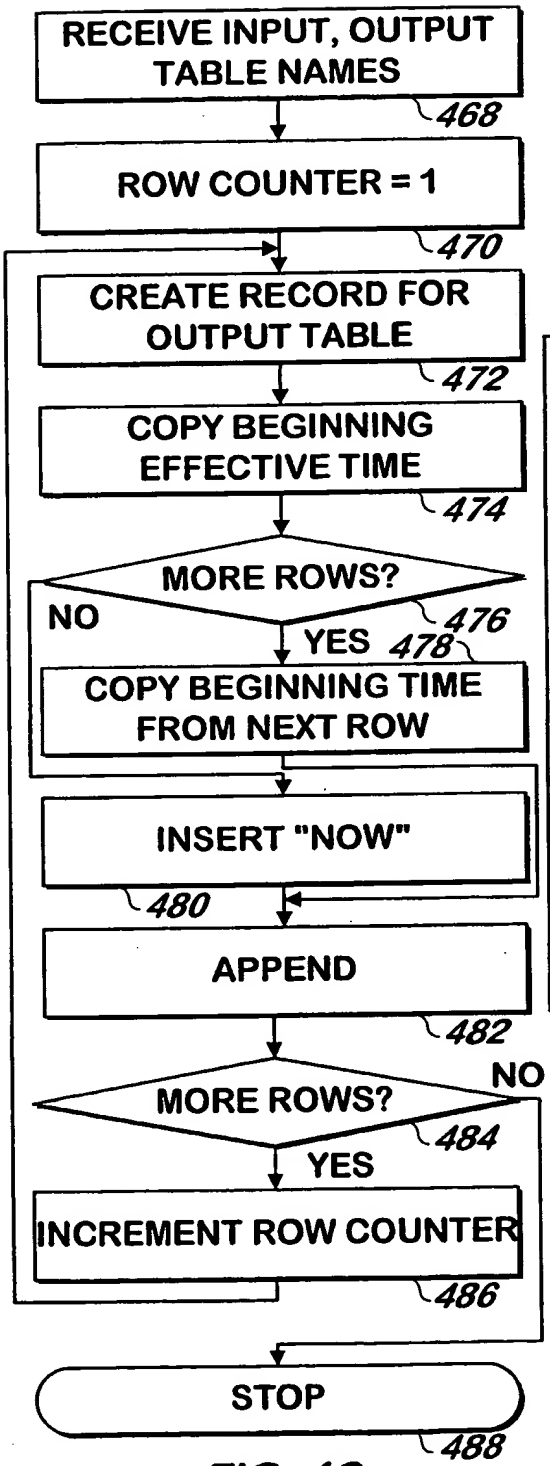

RECEIVE INPUT, OUTPUT TABLE NAMES
468
ROW COUNTER = 1
470
CREATE RECORD FOR OUTPUT TABLE
472
COPY BEGINNING EFFECTIVE TIME
474
MORE ROWS?
476
NO
YES
478
COPY BEGINNING TIME FROM NEXT ROW
INSERT "NOW"
480
APPEND
482
MORE ROWS?
484
NO
YES
INCREMENT ROW COUNTER
486
STOP
488

FIG. 4C

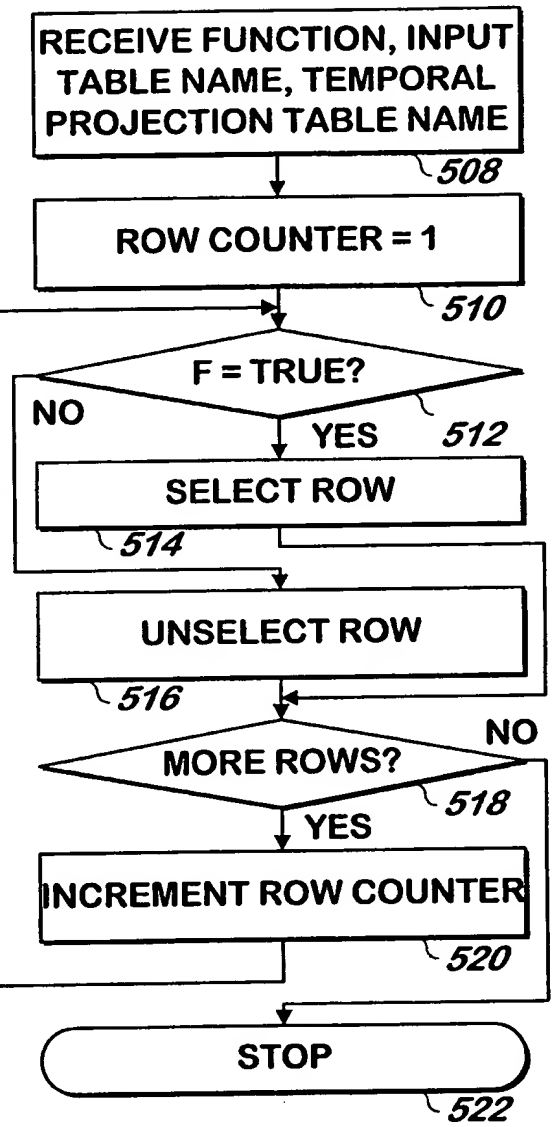

RECEIVE FUNCTION, INPUT TABLE NAME, TEMPORAL PROJECTION TABLE NAME
508
ROW COUNTER = 1
510
F = TRUE?
NO
YES
512
SELECT ROW
514
UNSELECT ROW
516
MORE ROWS?
518
NO
YES
INCREMENT ROW COUNTER
520
STOP
522

FIG. 5

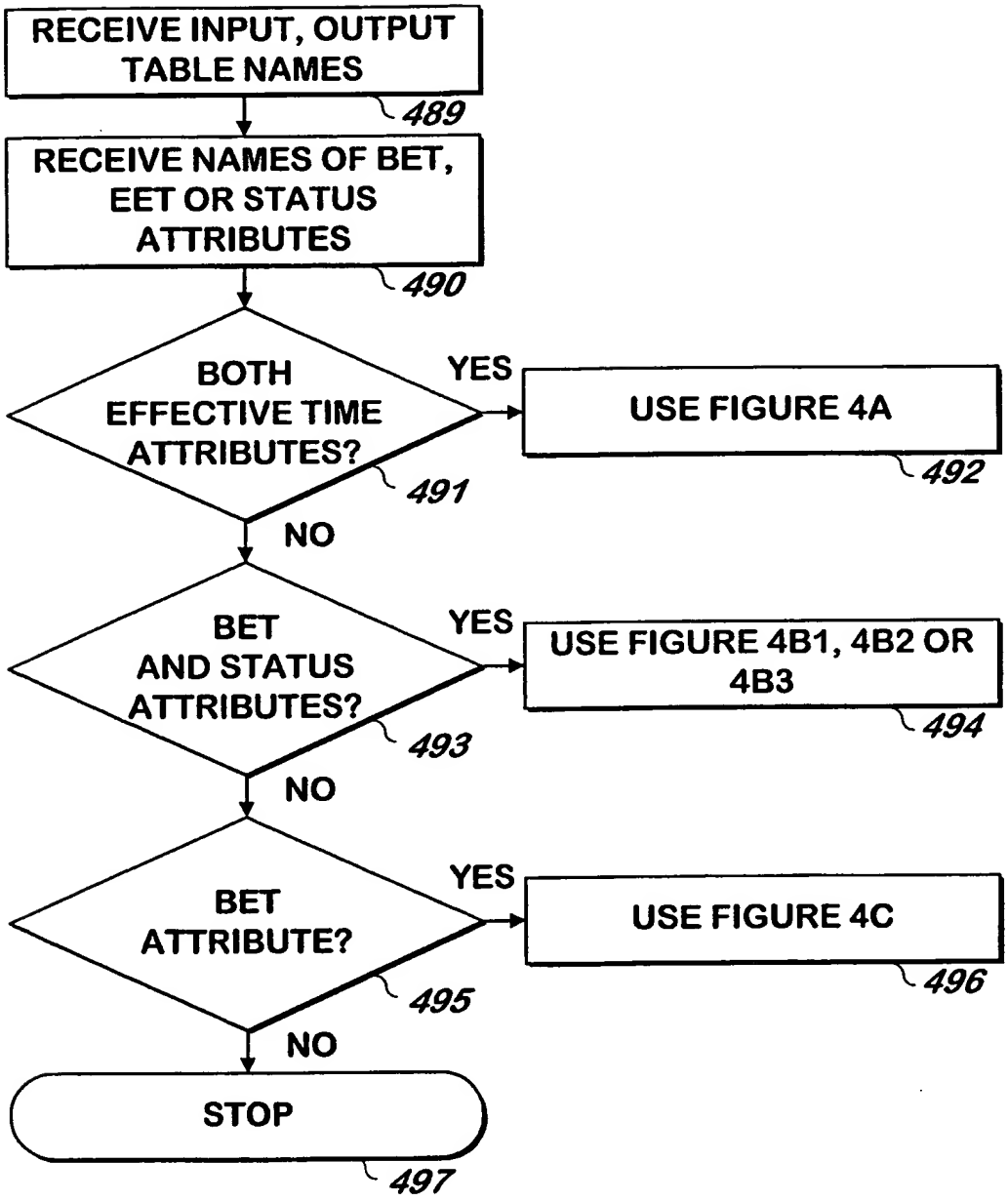

RECEIVE INPUT, OUTPUT TABLE NAMES
489
RECEIVE NAMES OF BET, EET OR STATUS ATTRIBUTES
490
BOTH EFFECTIVE TIME ATTRIBUTES?
YES
USE FIGURE 4A
492
491
NO
BET AND STATUS ATTRIBUTES?
YES
USE FIGURE 4B1, 4B2 OR 4B3
494
493
NO
BET ATTRIBUTE?
YES
USE FIGURE 4C
496
495
NO
STOP
497

FIG. 4D

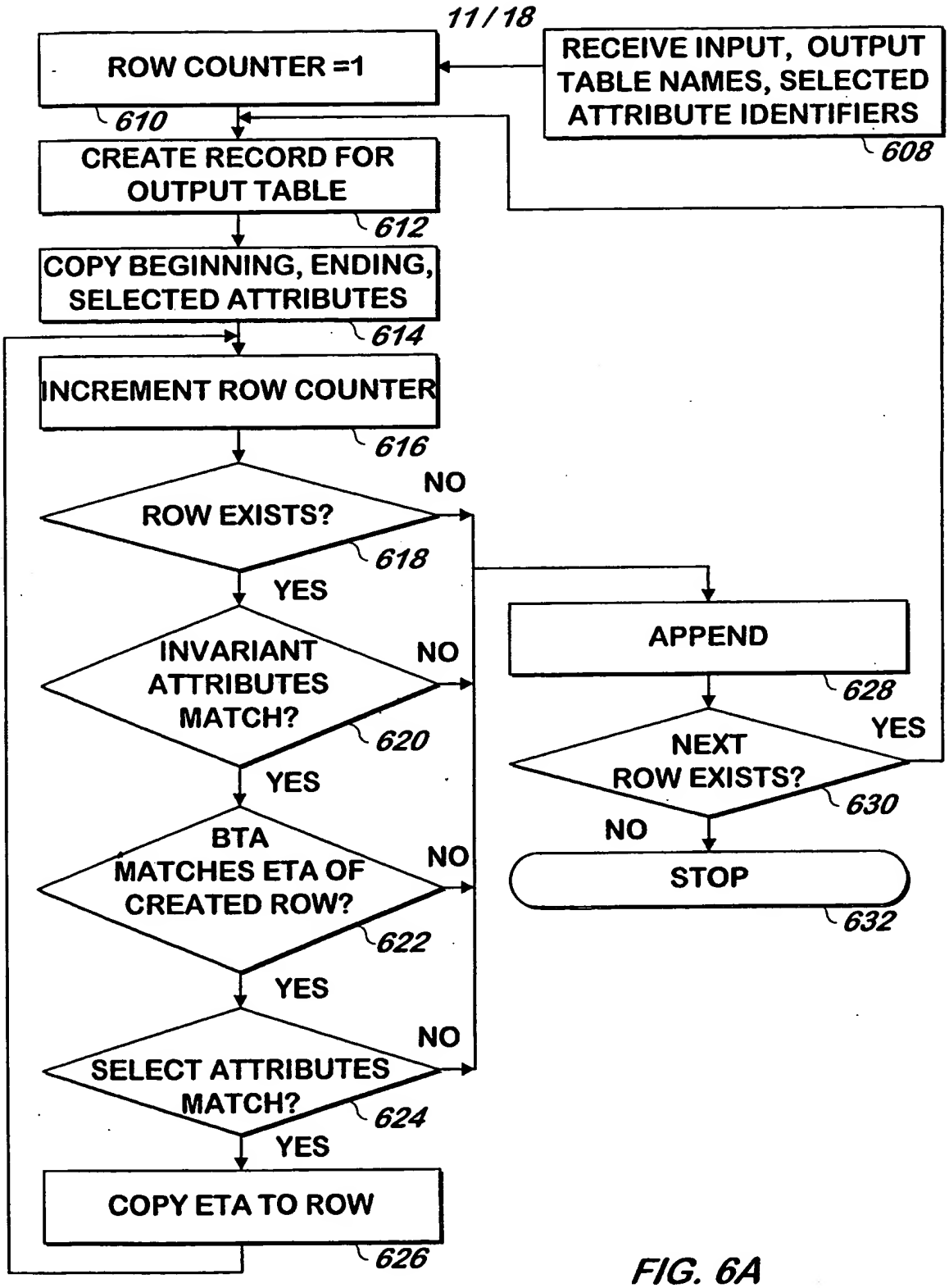

RECEIVE INPUT, OUTPUT TABLE NAMES, SELECTED ATTRIBUTE IDENTIFIERS
608
ROW COUNTER =1
610
CREATE RECORD FOR OUTPUT TABLE
612
COPY BEGINNING, ENDING, SELECTED ATTRIBUTES
614
INCREMENT ROW COUNTER
616
ROW EXISTS?
618
NO
YES
INVARIANT ATTRIBUTES MATCH?
620
NO
YES
BTA MATCHES ETA OF CREATED ROW?
622
NO
YES
SELECT ATTRIBUTES MATCH?
624
NO
YES
COPY ETA TO ROW
626
APPEND
628
NEXT ROW EXISTS?
630
YES
NO
STOP
632

FIG. 6A

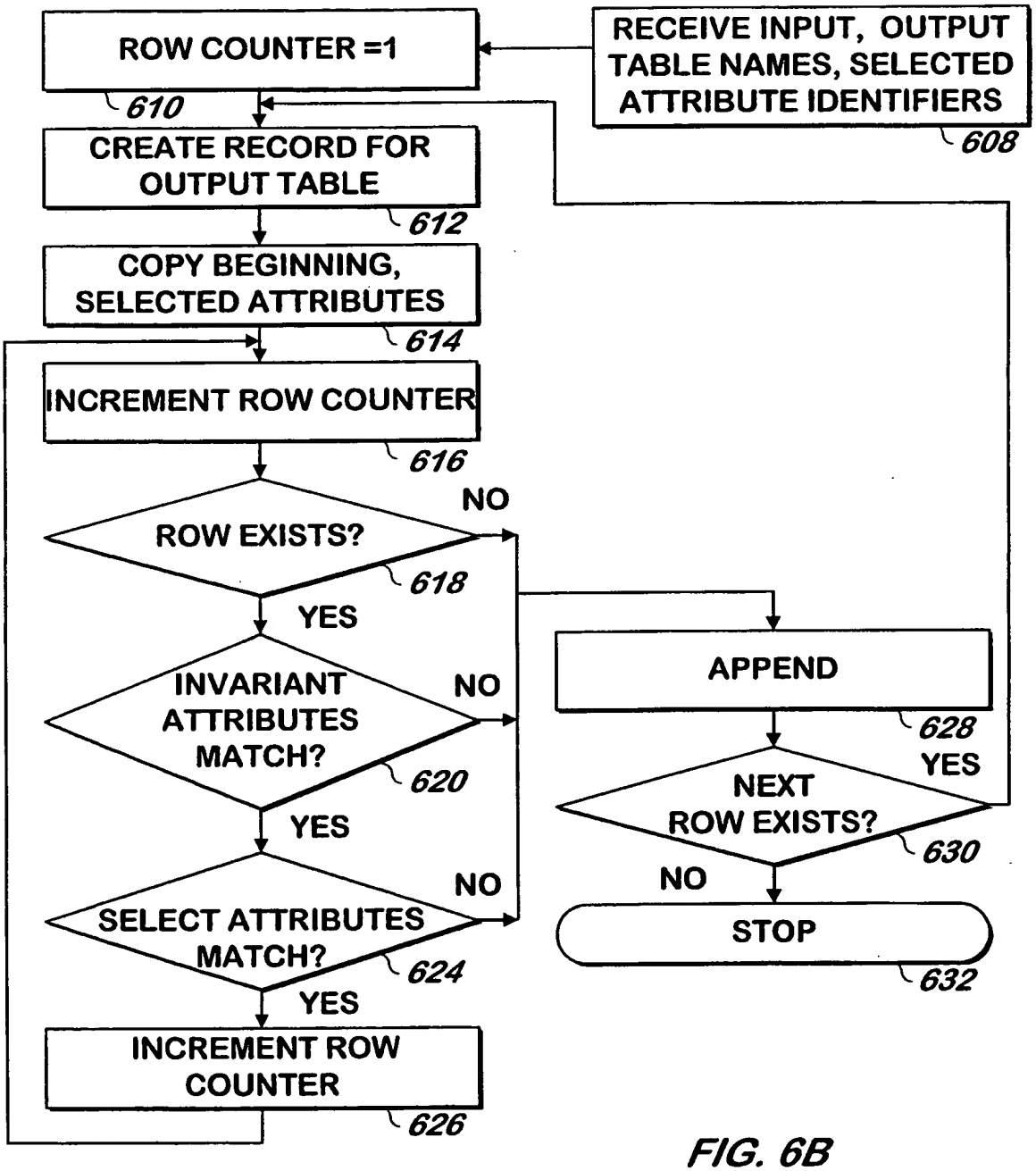

ROW COUNTER =1
610
RECEIVE INPUT, OUTPUT TABLE NAMES, SELECTED ATTRIBUTE IDENTIFIERS
608
CREATE RECORD FOR OUTPUT TABLE
612
COPY BEGINNING, SELECTED ATTRIBUTES
614
INCREMENT ROW COUNTER
616
ROW EXISTS?
618
NO
YES
INVARIANT ATTRIBUTES MATCH?
620
NO
YES
SELECT ATTRIBUTES MATCH?
624
NO
YES
INCREMENT ROW COUNTER
626
APPEND
628
NEXT ROW EXISTS?
630
YES
NO
STOP
632

FIG. 6B

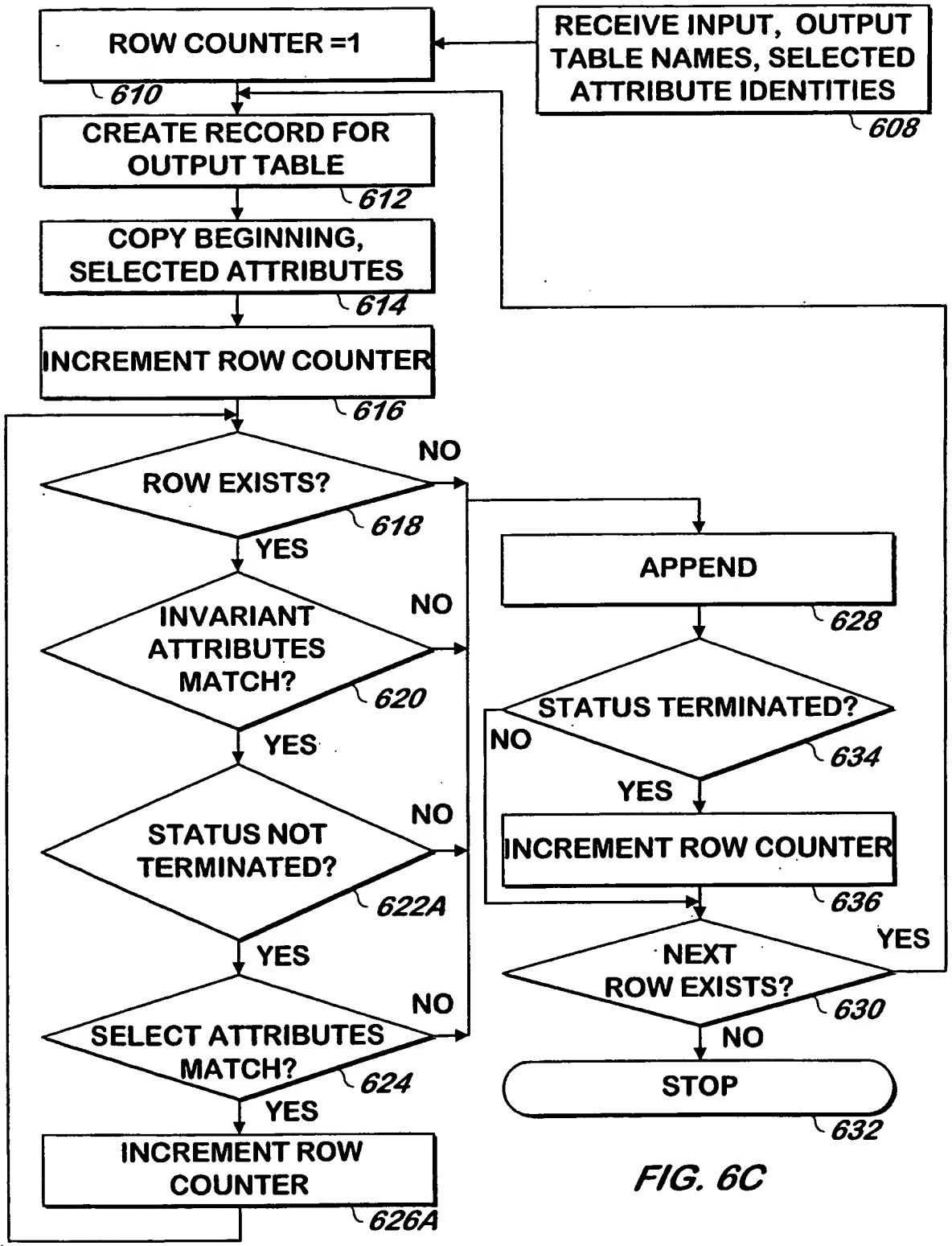

ROW COUNTER =1
610
RECEIVE INPUT, OUTPUT TABLE NAMES, SELECTED ATTRIBUTE IDENTITIES
608
CREATE RECORD FOR OUTPUT TABLE
612
COPY BEGINNING, SELECTED ATTRIBUTES
614
INCREMENT ROW COUNTER
616
ROW EXISTS?
618
NO
YES
INVARIANT ATTRIBUTES MATCH?
620
NO
YES
STATUS NOT TERMINATED?
622A
NO
YES
SELECT ATTRIBUTES MATCH?
624
NO
YES
INCREMENT ROW COUNTER
626A
APPEND
628
STATUS TERMINATED?
634
NO
YES
INCREMENT ROW COUNTER
636
NEXT ROW EXISTS?
630
YES
NO
STOP
632

FIG. 6C

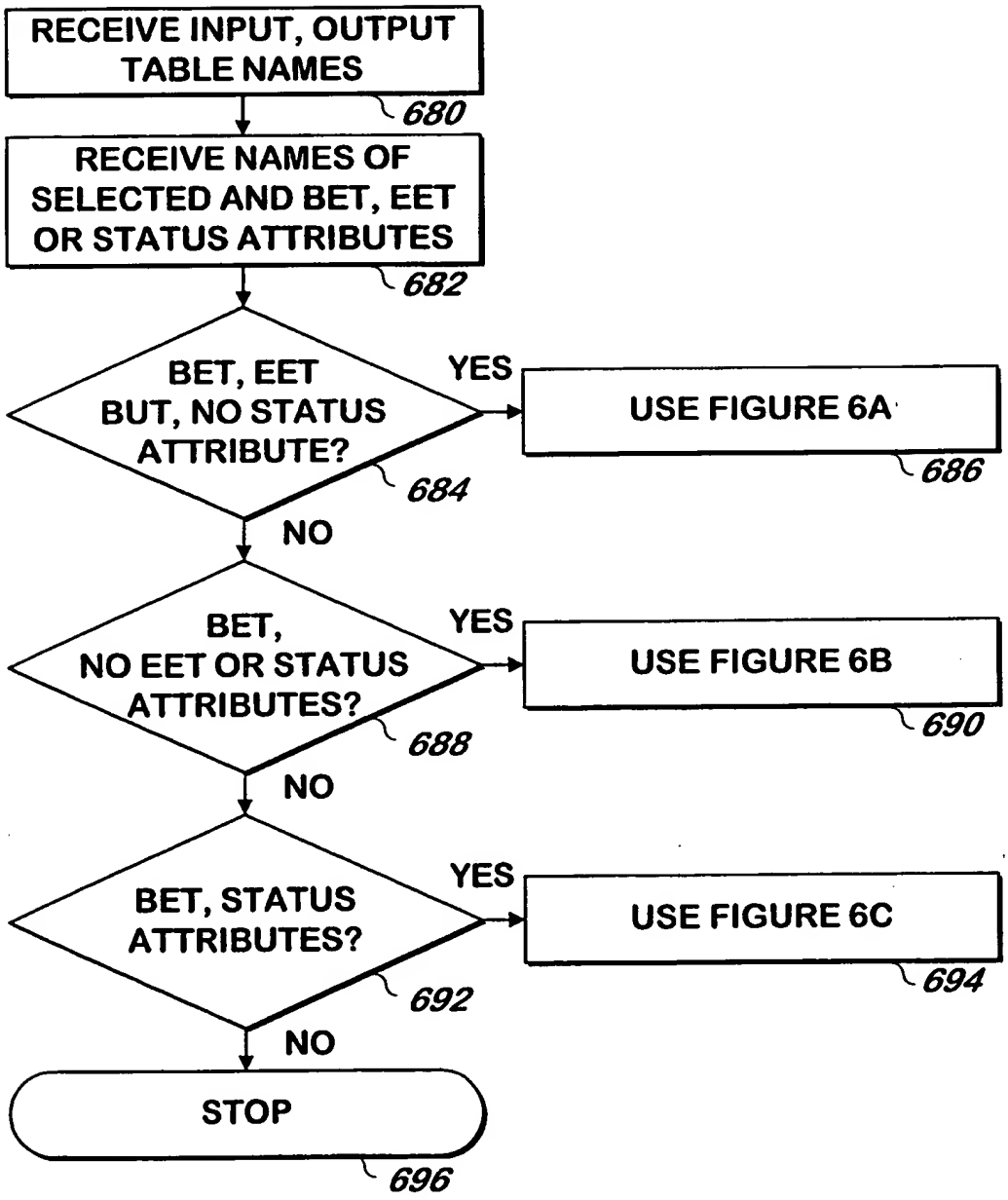

RECEIVE INPUT, OUTPUT TABLE NAMES
680
RECEIVE NAMES OF SELECTED AND BET, EET OR STATUS ATTRIBUTES
682
BET, EET BUT, NO STATUS ATTRIBUTE?
684
YES
USE FIGURE 6A
686
NO
BET, NO EET OR STATUS ATTRIBUTES?
688
YES
USE FIGURE 6B
690
NO
BET, STATUS ATTRIBUTES?
692
YES
USE FIGURE 6C
694
NO
STOP
696

FIG. 6D

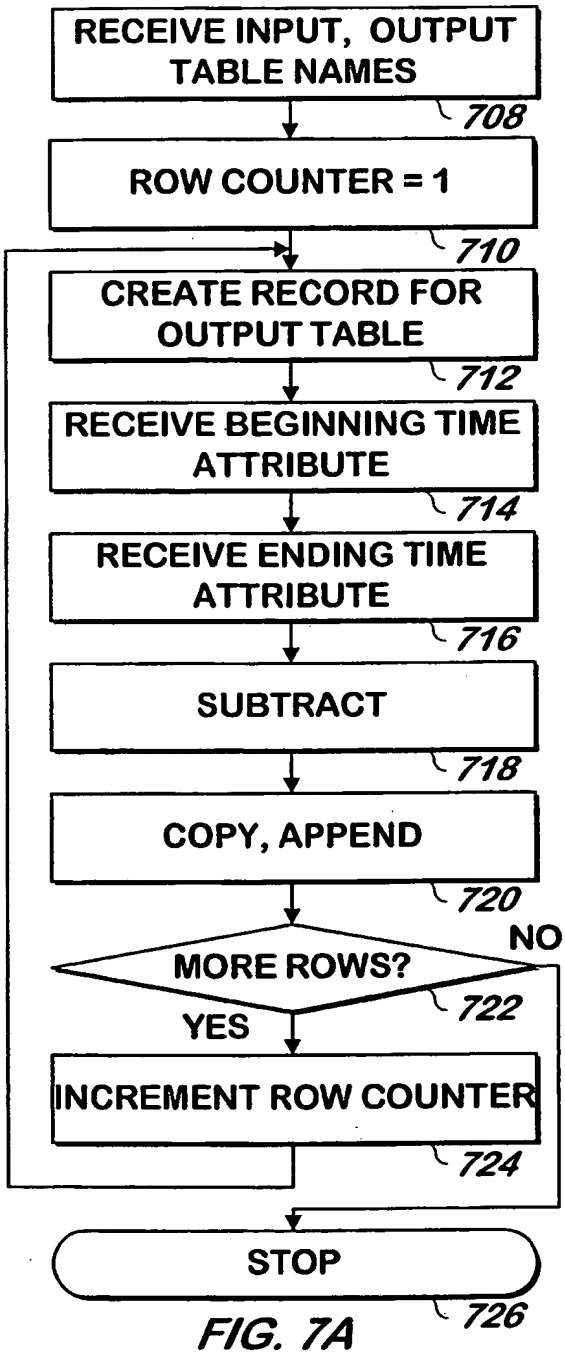

RECEIVE INPUT, OUTPUT TABLE NAMES
708
ROW COUNTER = 1
710
CREATE RECORD FOR OUTPUT TABLE
712
RECEIVE BEGINNING TIME ATTRIBUTE
714
RECEIVE ENDING TIME ATTRIBUTE
716
SUBTRACT
718
COPY, APPEND
720
MORE ROWS?
722
NO
YES
INCREMENT ROW COUNTER
724
STOP
726

FIG. 7A

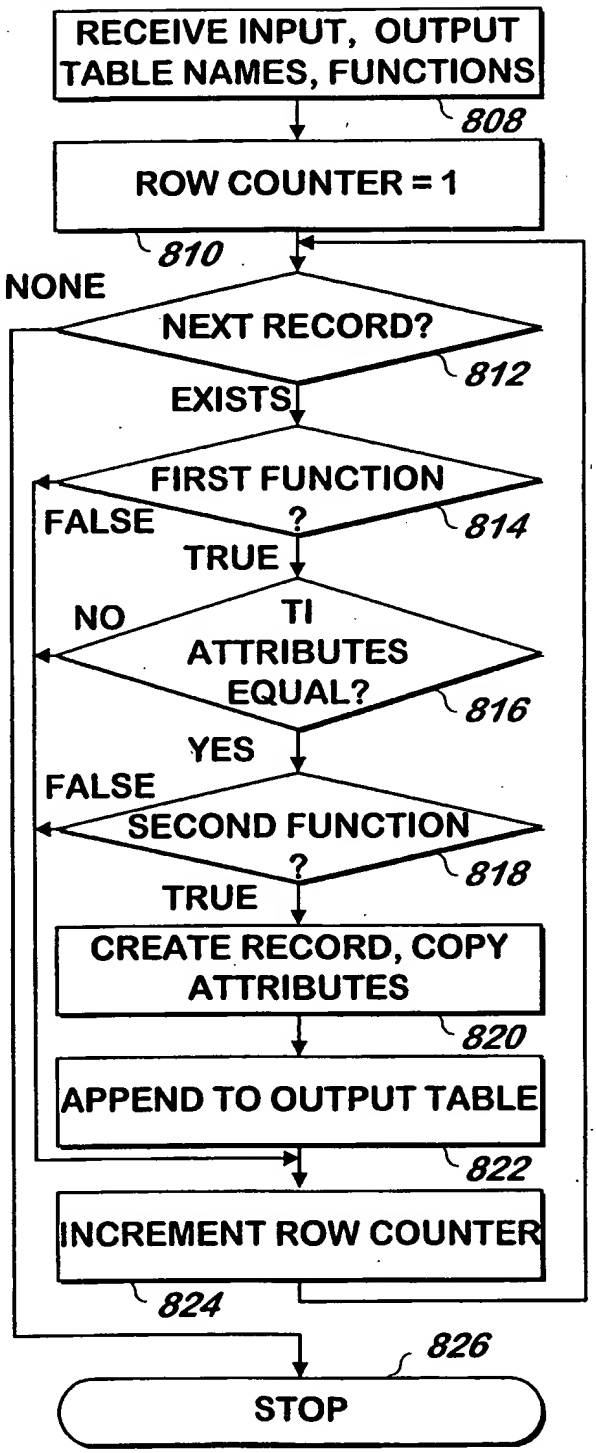

RECEIVE INPUT, OUTPUT TABLE NAMES, FUNCTIONS
808
ROW COUNTER = 1
810
NONE
NEXT RECORD?
812
EXISTS
FIRST FUNCTION ?
814
FALSE
TRUE
NO
TI ATTRIBUTES EQUAL?
816
YES
FALSE
SECOND FUNCTION ?
818
TRUE
CREATE RECORD, COPY ATTRIBUTES
820
APPEND TO OUTPUT TABLE
822
INCREMENT ROW COUNTER
824
826
STOP

FIG. 8A

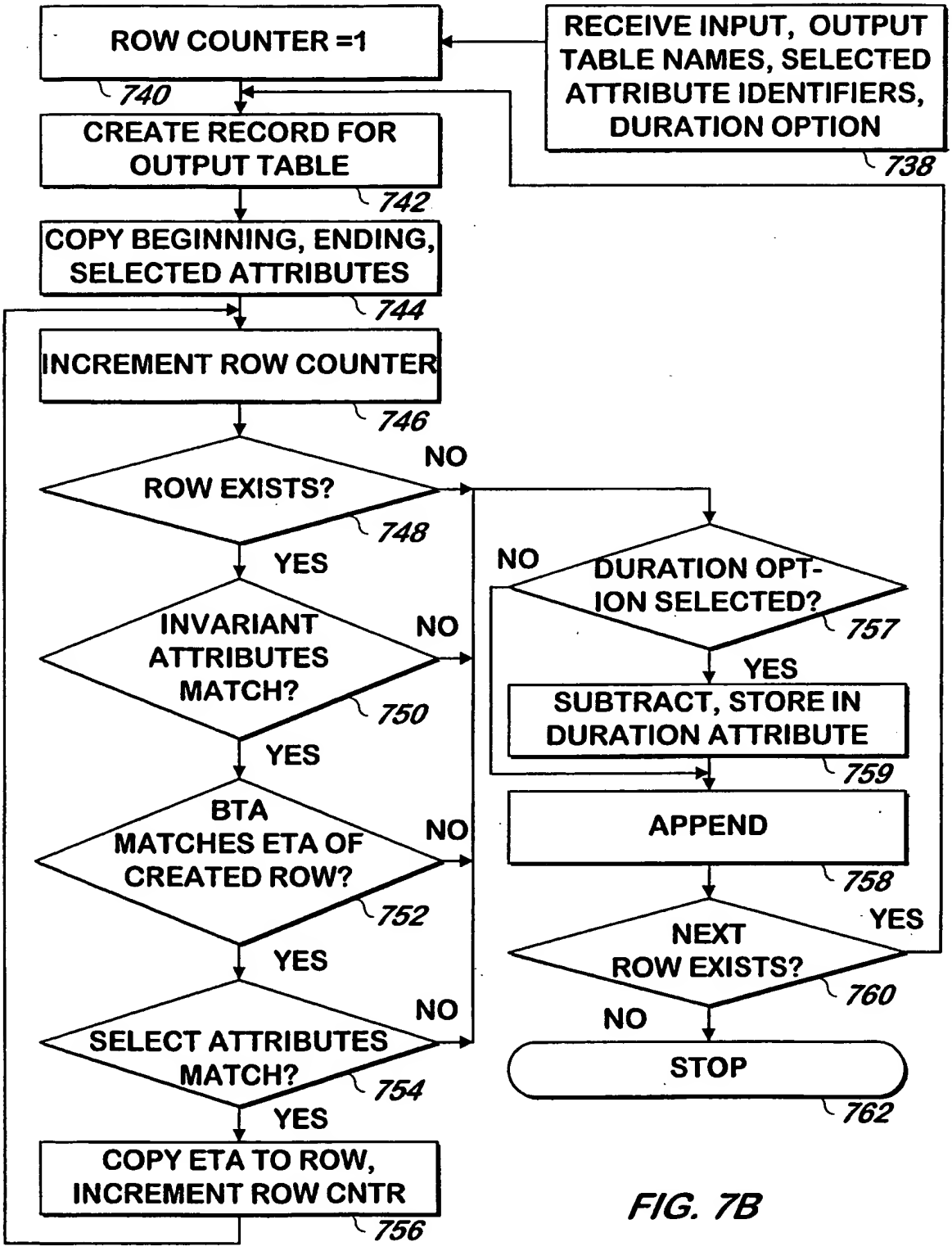

ROW COUNTER =1
740
RECEIVE INPUT, OUTPUT TABLE NAMES, SELECTED ATTRIBUTE IDENTIFIERS, DURATION OPTION
738
CREATE RECORD FOR OUTPUT TABLE
742
COPY BEGINNING, ENDING, SELECTED ATTRIBUTES
744
INCREMENT ROW COUNTER
746
ROW EXISTS?
748
NO
YES
INVARIANT ATTRIBUTES MATCH?
750
NO
YES
BTA MATCHES ETA OF CREATED ROW?
752
NO
YES
SELECT ATTRIBUTES MATCH?
754
NO
YES
COPY ETA TO ROW, INCREMENT ROW CNTR
756
DURATION OPTION SELECTED?
757
NO
YES
SUBTRACT, STORE IN DURATION ATTRIBUTE
759
APPEND
758
NEXT ROW EXISTS?
760
YES
NO
STOP
762

FIG. 7B

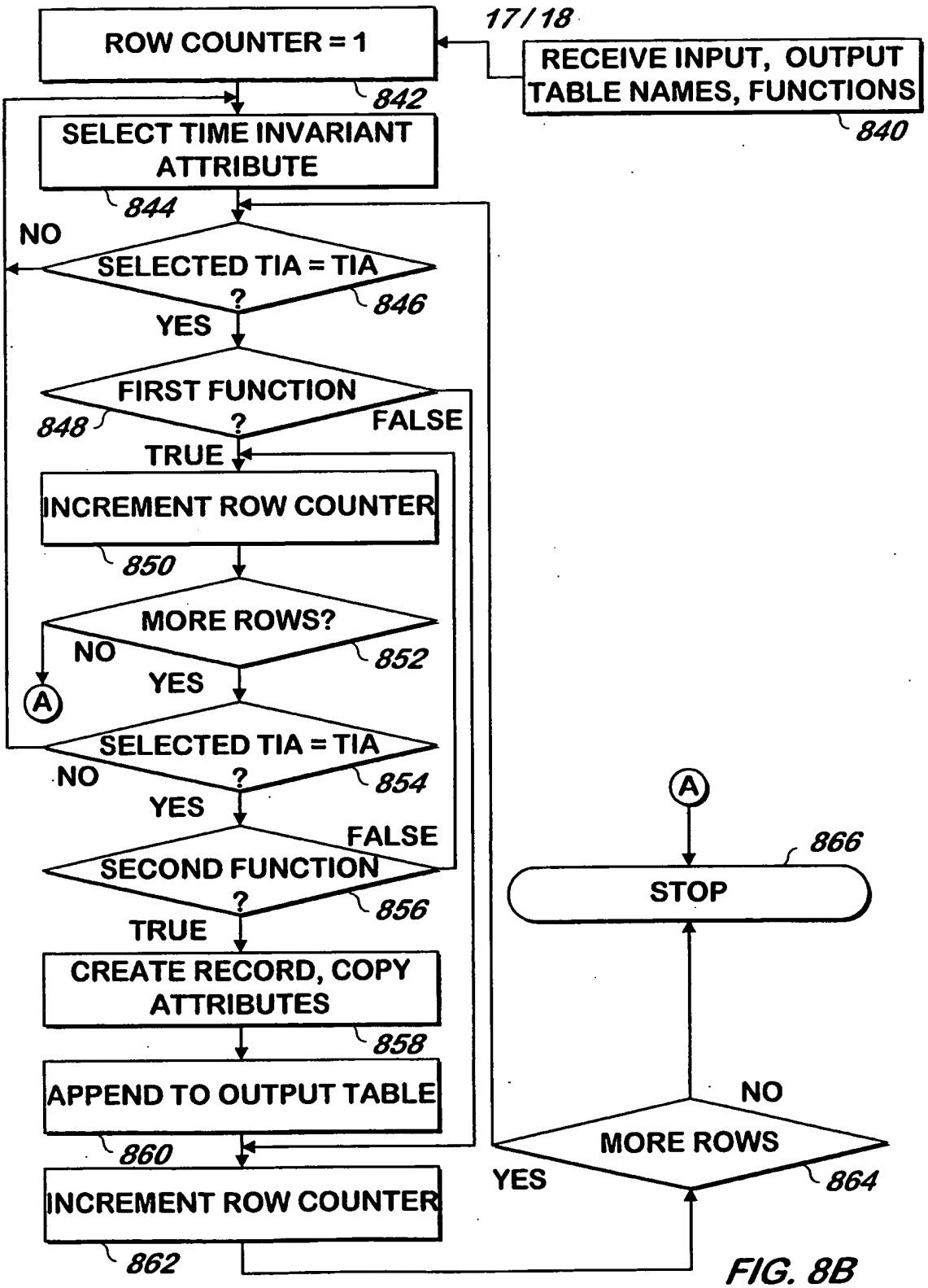

ROW COUNTER = 1
842
RECEIVE INPUT, OUTPUT TABLE NAMES, FUNCTIONS
840
SELECT TIME INVARIANT ATTRIBUTE
844
NO
SELECTED TIA = TIA ?
846
YES
FIRST FUNCTION ?
848
FALSE
TRUE
INCREMENT ROW COUNTER
850
MORE ROWS?
852
NO
YES
A
SELECTED TIA = TIA ?
854
NO
YES
FALSE
SECOND FUNCTION ?
856
TRUE
CREATE RECORD, COPY ATTRIBUTES
858
APPEND TO OUTPUT TABLE
860
INCREMENT ROW COUNTER
862
A
866
STOP
NO
MORE ROWS
864
YES

FIG. 8B

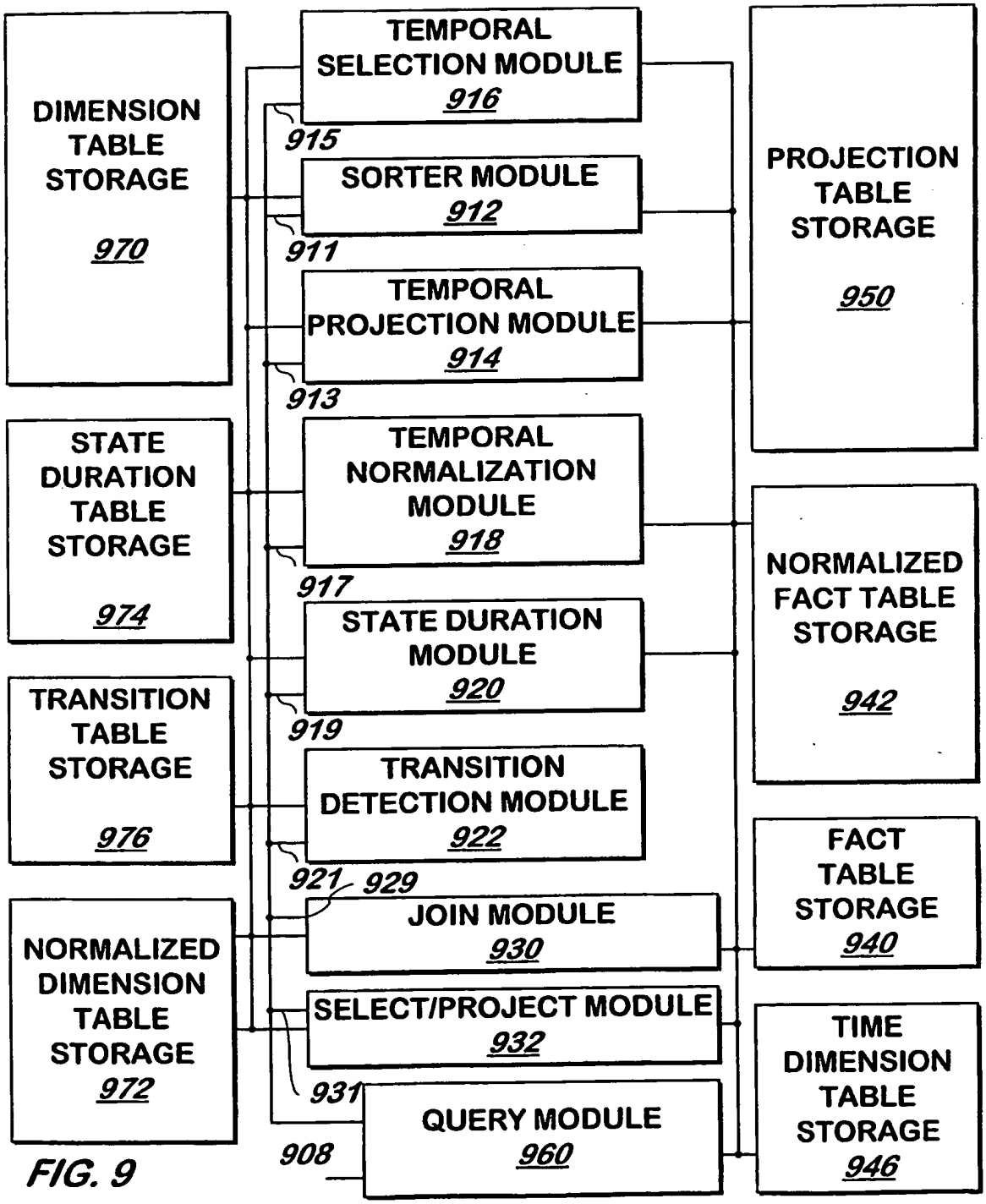

DIMENSION TABLE STORAGE
970
STATE DURATION TABLE STORAGE
974
TRANSITION TABLE STORAGE
976
NORMALIZED DIMENSION TABLE STORAGE
972
TEMPORAL SELECTION MODULE
916
915
SORTER MODULE
912
911
TEMPORAL PROJECTION MODULE
914
913
TEMPORAL NORMALIZATION MODULE
918
917
STATE DURATION MODULE
920
919
TRANSITION DETECTION MODULE
922
921
929
JOIN MODULE
930
SELECT/PROJECT MODULE
932
931
QUERY MODULE
960
908
PROJECTION TABLE STORAGE
950
NORMALIZED FACT TABLE STORAGE
942
FACT TABLE STORAGE
940
TIME DIMENSION TABLE STORAGE
946

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/23388

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 17/30
US CL : 707/3, 203
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/1. 3, 200, 203

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,440,730 A (ELMASRI et al.) 08 August 1995, col. 1, line 40 through col. 4, line 15.	1-31
Y	US 5,359,724 A (EARLE) 25 October 1994, col. 1, line 27 through col. 6, line 68.	1-31
A, P	US 5,745,755 A (COVEY) 28 April 1998, see entire document	1-31
A	FRIEND, D. Taking Databases into the Next Dimention, Computing Canada, Vol. 18, No. 20, page 40.	1-31

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
15 MARCH 1999	02 APR 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer JACK M. CHOULES Telephone No. (703) 305-9840

Form PCT/ISA/210 (second sheet)(July 1992) ★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/23388

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

DIALOG, APS, IEL Online, Faulkner Search.
Search terms: temporal, data, database, table, multidimennsion dimension, time.